ARTICLE IN PRESS

Information Sciences xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset

Dries Geebelen [a,*,1], Kristof Geebelen [b,1], Eddy Truyen [b], Sam Michiels [b], Johan A.K. Suykens [a], Joos Vandewalle [a], Wouter Joosen [b]

[a] ESAT-SCD-SISTA, Department of Electrical Engineering, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
[b] IBBT-DistriNet, Department of Computer Science, KU Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium

ARTICLE INFO

Article history:
Received 28 July 2011
Received in revised form 5 December 2013
Accepted 29 December 2013
Available online xxxx

Keywords:
QoS
SLA
Service composition
Time series prediction
Kernels
Online learning

ABSTRACT

Services offered in a commercial context are expected to deliver certain levels of quality, typically contracted in a service level agreement (SLA) between the service provider and consumer. To prevent monetary penalties and loss of reputation by violating SLAs, it is important that the service provider can accurately estimate the Quality of Service (QoS) of all its provided (composite) services. This paper proposes a technique for predicting whether the execution of a service composition will be compliant with service level objectives (SLOs). We make three main contributions. First, we propose a simulation technique based on Petri nets to generate composite time series using monitored QoS data of its elementary services. This techniques preserves time related information and takes mutual dependencies between participating services into account. Second, we propose a kernel-based quantile estimator with online adaptation of the constant offset to predict future QoS values. The kernel-based quantile estimator is a powerful non-linear black-box regressor that (i) solves a convex optimization problem, (ii) is robust, and (iii) is consistent to the Bayes risk under rather weak assumptions. The online adaption guarantees that under certain assumptions the number of times the predicted value is worse than the actual value converges to the quantile value specified in the SLO. Third, we introduce two performance indicators for comparing different QoS prediction algorithms. Our validation in the context of two case studies shows that the proposed algorithms outperform existing approaches by drastically reducing the violation frequency of the SLA while maximizing the usage of the candidate services.

* Corresponding author. Tel.: +32 16/328656.
E-mail addresses: dries.geebelen@esat.kuleuven.be (D. Geebelen), kristof.geebelen@cs.kuleuven.be (K. Geebelen), eddy.truyen@cs.kuleuven.be (E. Truyen), sam.michiels@cs.kuleuven.be (S. Michiels), johan.suykens@esat.kuleuven.be (J.A.K. Suykens), joos.vandewalle@esat.kuleuven.be (J. Vandewalle), wouter.joosen@cs.kuleuven.be (W. Joosen).
[1] Both authors contributed equally to this work.

http://dx.doi.org/10.1016/j.ins.2013.12.063
0020-0255/

1. Introduction

1.1. Context

Workflow languages, such as WS-BPEL,[2] focus on combining web services into aggregate services that satisfy the needs of clients. A service composition consists of a collection of related, structured activities or tasks that produce a specific service by combining services provided by multiple business partners. Tasks can be delegated to globally available software services and may require human interaction. For example, an integrated travel planning web service can be created by composing services for hotel booking, airline booking, payment, etc.

In a global service market, many available web services can provide similar functionality, but have different Quality of Service (QoS). QoS can be defined in terms of attributes such as price, response time, availability and reputation [4]. QoS-aware service composition refers to the process of composing services in function of their QoS properties such that the overall composition meets the QoS constraints, specified in the service level agreement (SLA) [3]. Violating SLAs is often associated with a decrease in reputation or monetary penalties for the service provider [20]. It is therefore in the providers' best interest to make for each potential composition an accurate assessment of the QoS that will be provided during its execution by the client. In this context, two important challenges arise: *QoS aggregation* [11,10,17] and *QoS prediction* [5,21,15]. Composite services typically consist of different activities such as sequences, conditions, loops and parallel invocations. To calculate the aggregated QoS of the composition, these different composition patterns have to be taken into account. Furthermore, web services typically operate autonomously within a rapidly changing environment. As a result, their QoS may change frequently, either because of internal changes or because of changes in their environment [31]. To know if a composition will respect the SLA during its execution, the provider must thus predict what the QoS of the composition will be during its actual execution by the client.

1.2. Problem statement

The problem tackled in this paper is to produce future QoS values of the composition as accurate as possible given historical QoS values of the individual services. A selected composition can then be rejected or accepted to address the customers' requests depending on whether the prediction indicates if the composition will be executed according to the service level objectives (SLOs) of the SLA. In the context of this paper, an SLO states that an accepted service should satisfy a QoS constraint with a probability greater than τ (e.g. the service must respond in less than 1 s in 99,9% of the cases). Two types of errors can occur in this scenario: a type I error occurs when the service is rejected and would have satisfied the SLO, a type II error occurs when the service is accepted and will violate the objective. Suppose f_τ is a function that predicts a future QoS attribute given a certain input (e.g. past values of that QoS attribute) and f_τ belongs to a certain hypothesis space $\mathcal{H}$. If based on this prediction we decide whether or not to accept the request (e.g. we accept if the predicted value is below or above a certain threshold value), then the problem we try to optimize is

$$\min_{f_\tau \in \mathcal{H}} P(\text{Type I error}|f_\tau)(1-\tau) + P(\text{Type II error}|f_\tau)\tau \tag{1a}$$

$$\text{such that } P(\text{Type II error}|\text{accepted}, f_\tau) \leqslant 1-\tau \tag{1b}$$

where P expresses a probability and 'accepted' means the service was accepted to execute the task. Constraint (1b) states that the frequency of accepted services that violate the SLO should not exceed $1-\tau$. As long as constraint (1b) is satisfied, we have chosen the cost we want to minimize in Eq. (1a) to be linear in the number of type I and type II errors: an error of type I has cost $1-\tau$ and an error of type II has cost τ. We will show in this paper that in order to minimize optimization problem (1), we need to know the conditional τ-quantile value. From an economical point of view it is important to accept as many requests as possible, while the proportion of violations of the accepted request should be as small as possible. Therefore it is interesting to reformulate problem (1) as an optimization problem in the rejection rate (P(rejected)) and the violation rate (P(Type II error|accepted))

$$\min_{f_\tau \in \mathcal{H}} P(\text{Type II error}|\text{accepted}, f_\tau) + P(\text{rejected}|f_\tau)((1-\tau) - P(\text{Type II error}|\text{accepted}, f_\tau)) \tag{2a}$$

$$\text{such that } (1-\tau) - P(\text{Type II error}|\text{accepted}, f_\tau) \geqslant 0 \tag{2b}$$

where *rejected* means the service was rejected to execute the task. A proof for this reformulation can be found in Appendix A. Given constraint (2b) holds, the cost we want to minimize decreases for a decreasing rejection rate and a decreasing violation rate as desired.

[2] Web Services Business Process Execution Language Version 2.0, April 2007, OASIS Technical Committee, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

1.3. Related work

With respect to *QoS aggregation*, the current state-of-the-art techniques have some important drawbacks. As explained in [21], some existing approaches for aggregation use *hard* composition rules such as addition, maximum or conjunction [4,5,31,1,3,9,30]. In case one is interested in quantile values, hard contract rules can be overly pessimistic because they do not take the probability distribution of the individual services into account. This problem can be solved by estimating probability distributions for the QoS values of the elementary services which are combined according to *soft* composition rules [10,21]. Soft contract rules, on the other hand, assume that QoS values of the different elementary services are independently distributed. In practice, however, quality attributes such as response time of different automated services are often dependent due to several reasons: both services run on the same server; during the day certain services are executed more often than during the night; and users choose the fastest service out of a group of services which causes the services to become equally fast. These dependencies are even more distinct with non-automatic services that require human interaction to execute a task. They are often induced due to the fact that outside working hours less people are available to execute a task.

Concerning *QoS prediction* it is important that the predicted quantile value approximates the true conditional quantile function as good as possible. We did not find any seminal work that uses state-of-the-art prediction methods to predict quantile values in this context.

1.4. Contributions

This paper has the following contributions. First, we present a *QoS aggregation* technique that generates composite time series using monitored QoS data of its elementary services. Our proposed simulation technique based on Petri nets preserves time related information and takes mutual dependencies between participating services into account. Second, we address *QoS prediction* by solving optimization problem (1) using a kernel-based quantile regressor with online adaptation of the constant offset. This estimator has the following properties:

- It can guarantee that, under certain assumptions, the number of times the predicted value is worse than the actual value converges to the agreed quantile value for the number of data points going to infinity.
- Kernel-based methods like SVM [27] and LS-SVM [24] have shown to be powerful non-linear black-box methods successful for various applications such as optical character recognition [16] and electricity load prediction [6].
- The solved optimization problem is convex as opposed to other non-linear black-box methods such as neural networks. This guarantees the global optimum can be found efficiently.
- It is robust and thus resistant to outliers. Its breakdown point equals $1 - \tau$ if $\tau \geqslant 0.5$.
- It is risk consistent to the Bayes risk under rather weak assumptions and it approximates the conditional quantile function [7].

Third, we introduce two performance indicators to quantify and compare our results: the first expresses, given certain assumptions, the cost we want to minimize in (1a) and the second expresses, given certain assumptions, the likelihood (1b) holds.

1.5. Structure of the paper

This paper is organized as follows: Section 2 clarifies the problem statement and motivates our approach. Section 3 elaborates on related work. Section 4 provides an overview of the QoS model we use for this work. We propose the simulation technique based on Petri nets for calculating the QoS attributes of composite services, given QoS attributes of its elementary services in Section 5. Section 6 explains the performance indicators and describes the underlying prediction mechanism that is used to predict violations of the SLA between customer and service provider. An experimental evaluation of our approach and comparison with existing work is documented in Section 7. Section 8 provides a conclusion. Finally, Section 9 elaborates on future work.

2. Motivation

2.1. Running example

This section presents a case study situated in the health care environment. The case study consists of a composite service (workflow), initiated by the government, that realizes a mammography screening program in order to reduce breast cancer mortality. The workflow is illustrated in Fig. 1.

The first task of the workflow consists of sending out invitations to all women that qualify for the program. A radiologist will take images needed for screening and uploads them to the system (task 2). Next, the images need to be analysed by specialized screening centers. There are always two independent readings, represented by tasks 3 and 4. These readings

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

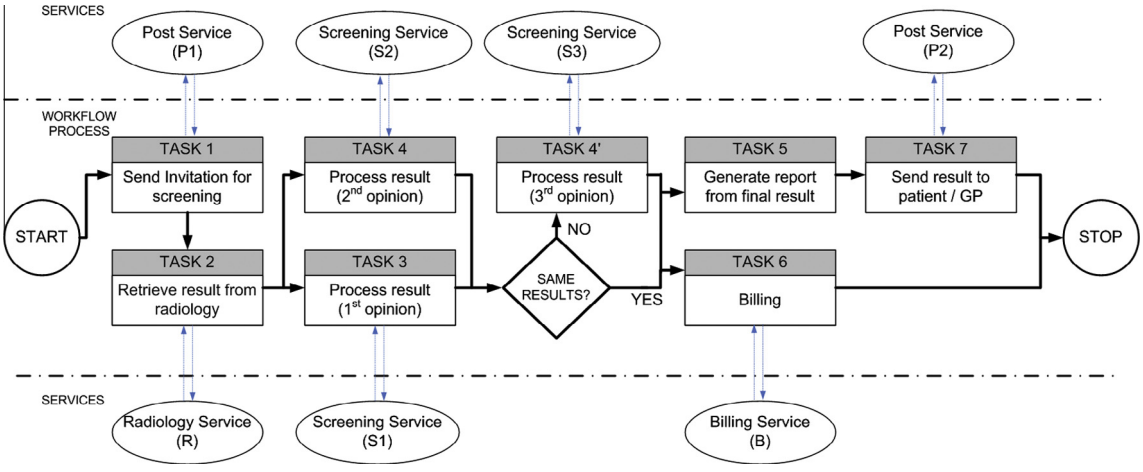

Fig. 1. Example e-health workflow process.

can be performed in parallel. In a next step, the two results of the readings are compared. When the results are identical, it is unlikely that the two physicians made the same mistake. Therefore it can be safely assumed that results are correct and the workflow can proceed with task 5. However, when the results are different, a concluding reading is performed (task 4′). Once the results of the screening of a particular screening subject are formulated, a report is generated (task 5) and a report is sent to the screening subject and her general practitioner in task 7. In parallel, different parties are billed (task 6).

Suppose the government, who finances this initiative, wants some quality guarantees and specifies a service level agreement with the company (service provider) responsible for executing the workflow. In this agreement, the company specifies, for example, that in 99% of the cases the duration of the workflow composition will take no longer than 15 working days. We will now motivate why quantile estimation, time series prediction and taking into account mutual QoS dependencies are important for accurate SLO compliance estimation.

2.2. Quantile vs. average SLO compliance estimation

The algorithms we propose are based on quantile estimation. We explain its benefit by means of a simple example. Suppose two different service selections on the e-health workflow leads to two composite services CS1 & CS2. The response time of these services have, at a certain time, probability densities for RTs of 0–25 days as presented in Table 1. We can use these values to make a quality estimate for an SLO. For example, which service would be the best candidate to have 99% certainty that its response time will not exceed 15 days? Using the average RTs of 12.5 days for CS1 and 7.5 days for CS2, it seems that CS2 is more reliable than CS1 and thus the best choice. However, using 99% quantile values of 15 days for CS1 and 20 days for CS2, we rightly conclude the opposite: CS1 has a higher probability that it will not exceed the threshold of 15 days because it is less volatile.

2.3. Time series prediction

Again, consider a scenario where one has to estimate which of two composite services, as shown in Table 2, is the best candidate to comply with an SLO stating that the total duration of the composite service will take no longer than 15 working days. If we expand the pattern present in $t1, t2, t3$ and $t4$ to $t5$ and $t6$ as shown in Table 2, then at $t5$ 'CS A' complies with the SLO while 'CS B' violates the SLO and at $t6$ the opposite happens. The estimation for this scenario, as for many real-life scenario's, is thus time dependent. Possible causes for varying quality of service attributes are temporary over -or underload, infrastructure failures, seasonality due to fixed working hours of non-automatic services, etc. The algorithms we propose to predict time varying QoS attributes are discussed in Section 6.

Table 1
Probability density, average and 99%-quantile values of RTs for 2 composite services CS1 & CS2.

Services	Monitored RT (Prob. Density)					99% Q-value	Average value
	0–5 (%)	5–10 (%)	10–15 (%)	15–20 (%)	20–25 (%)		
CS 1	0	1	98	1	0	15	12.5
CS 2	15	75	6	3	1	20	7.5

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

Table 2
Past (monitored) and future (to be predicted) RTs for two composite services.

Services	Monitored RT (days)				Real RT (days)	
	t1	t2	t3	t4	t5	t6
CS A	10	25	10	25	10	25
CS B	20	10	20	10	20	10

2.4. Mutual dependencies

In this example we will show why taking into account mutual dependencies (e.g. as a consequence of time dependencies) is important. Suppose the composite service (CS) consists of a screening services and a post service (SS and PS) executed in sequence as shown in Table 3. The response times are dependent in the sense that if the execution of SS contains a weekend, then the execution of PS does not contain a weekend. The true worst-case response time is 5 days (3 working days and an extension of 2 days due to the weekend). Algorithms assuming independence of response times will estimate the worst-case response time as 7 days (3 days for SS plus 4 days for PS) because they add the penalty due to weekends twice. Using these values, for example to calculate the 99% QoS quantile, will result in a too pessimistic value (7 days instead of 5 days), causing a competitive disadvantage to the service provider. This example shows why in real scenario's assuming independence can be dangerous. Mutual dependencies as a consequence of sharing resources or other dependencies give similar results.

3. Related work

In literature, several works can be found that address QoS-aware service composition. Surveys are reported in [12,29,23]. Important research challenges in this domain are **QoS aggregation, QoS-based service selection** and **QoS prediction**. Fig. 2 and Table 4 give an overview of related work for each challenge. We discuss them briefly.

QoS aggregation, which is typically part of service selection and prediction, involves calculating a global QoS value based on the QoS of the participating services. Jaeger et al. [11] introduce an aggregation method to calculate the QoS of the composition, given the QoS values of the individual services. The model identifies several structural elements called composition patterns. These structural elements were derived from a set of workflow patterns defined in Van der Aalst [26]. A practical approach is taken by Mukherjee et al. [17]. They propose a model for estimating three key QoS parameters (Response Time, Cost and Reliability) of an executable BPEL process from the QoS information of its partner services and certain control flow parameters. Hwang et al. [10] approaches the aggregation challenge as a stochastic problem. Each QoS parameter for a web service is represented as a discrete random variable with a probability mass function. They propose a probabilistic framework to derive a QoS measure of a composite service from those of its constituent services and explore algorithms for computing the probability distribution functions of the QoS of the service composition. Their theoretical rules for composing the probability mass function are based on the assumption that QoS values of each constituent web service of a composition construct are mutually independent.

QoS-based service selection deals with finding an assignment of services to workflow tasks which maximizes a customer related utility function. Typically this comes down to the following optimization problem: given an abstract composite service and a set of candidate services with different QoS values for each task, find a service for each task such that the utility is maximized and the global composite QoS values satisfy certain Service Level Objectives (SLOs). Popular techniques in literature to solve this challenge efficiently are integer programming [31,1], efficient heuristic algorithms [30] and genetic algorithms [3]. These works tackle the composition problem assuming deterministic QoS attributes for the elementary services. Harney and Doshi [9] present a composition solution that intelligently adapts workflow processes to changes in quality parameters of participating services providers. Their approach assumes that QoS values remain fixed for a certain amount of time. Changes are introduced by means of expiration times, i.e. service providers provide their current reliability rates and duration of time for which the current reliability rates are guaranteed to remain unchanged. Wiesemann et al. [28] focus on the stochastic service composition problem. They formulate the service composition problem as a multi-objective stochastic program which simultaneously optimizes QoS parameters which are modeled as decision-dependent random

Table 3
Monitored response times for the screening service (SS), the post service (PS) and the corresponding response time for the composite service. The response time for SS equals one working day and the response time for PS equals 2 working days. During the weekend (when no personnel is available to execute the tasks) the progress freezes and the response time will be extended by up to 2 days.

	Mo	Tu	We	Th	Fr	Sa	Su	**(Max)**
SS	1	1	1	1	3	2	1	3
PS	2	2	2	4	4	3	2	4
CS	3	3	5	5	5	4	3	5

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

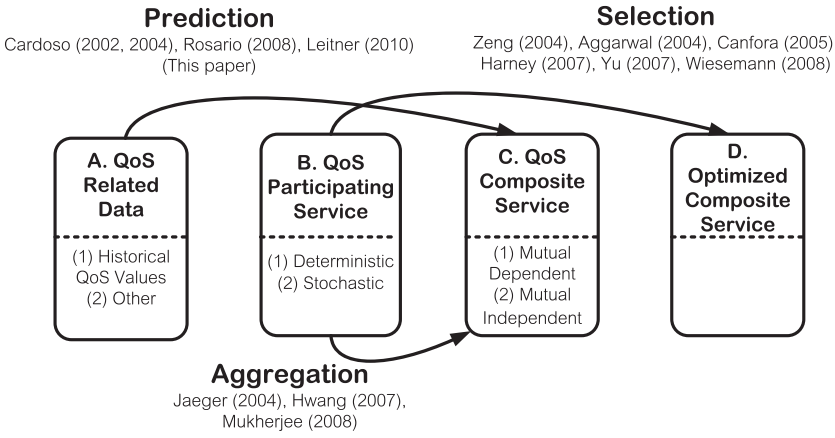

Fig. 2. Related work – positioning.

Table 4
Related work – overview.

Related work	Challenge	Approach	Implementation choices
Cardoso [4,5]	Prediction	Stochastic workflow reduction	Hqv (avg, min, max), mutually independent
Jaeger et al. [11]	Aggregation	Composition patterns	Deterministic
Zeng et al. [31]	Selection	Linear integer programming	Deterministic
Aggarwal et al. [1]	Selection	Linear integer programming	Deterministic
Canfora et al. [3]	Selection	Genetic algorithms	Deterministic
Hwang et al. [10]	Aggregation	Probabilistic composition patterns	Stochastic, mutually independent
Harney and Doshi [9]	Selection	Value of changed information	Deterministic
Yu et al. [30]	Selection	Efficient heuristic algorithms	Deterministic
Wiesemann et al. [28]	Selection	Stochastic programming	Stochastic, mutually independent
Rosario et al. [21]	Prediction	Monte Carlo sampling	Hqv (distribution), mutually independent
Mukherjee et al. [17]	Aggregation	Composition patterns for WS-BPEL	Deterministic, mutually independent
Leitner et al. [15]	Prediction	Regression	Other (runtime data of composition)
This paper	Prediction	Petri nets, Support vector machines	Hqv (time series), mutually dependent

Hqv: Historical values of qos attributes
Deterministic: Qos values of participating services are fixed point estimates
Stochastic: Qos values of participating services are stochastic variables that change in time
Mutually independent: Qos values among participating services do not depend on each other

variables. Their model minimizes the average value-at-risk (AVaR) of the workflow duration and costs while imposing constraints on the workflow availability and reliability.

Driven by the fact that QoS attributes, such as response time, can be very volatile with respect to time, the challenge of **QoS prediction** has recently gained popularity. Its main goal is to bridge the time gap between the service selection process and the actual execution of the composite service. By predicting accurate expected values for quality measures in the near feature, this technique is able to improve the probability that the selected composition of services will still respect the SLO during the execution of the workflow. This challenge typically starts from QoS related data and assumes that a selection of services for the composition has taken place. Cardoso's Ph.D. thesis [4] is a seminal work that proposes a framework that uses Stochastic Workflow Reduction to arrive at QoS estimates for the overall workflow. Using different workflow patterns, they defined QoS aggregation with four attributes: response time, cost, reliability, and fidelity. These aggregation patterns make it possible to predict the QoS performance of service processes by performing the substitution repeatedly until the whole process is transformed into a composite service node. Although Cardoso mentioned the possibility of deriving distribution functions for QoS of workflow tasks, the proposed reduction rules were only implemented for point estimates, such as minimum, average, and maximum, of historical QoS values. Leitner et al. [15] present a framework called PREvent for event-based monitoring and prediction of SLA violations and automated runtime prevention by triggering adaptation actions in service compositions. Monitored runtime data of the composition is used by a predictor to identify problematic instances at defined checkpoints in the composition execution via regression. Because their approach does not use historical QoS values of the participating services, there is no need for aggregation or for taking service dependencies into account. Rosario et al. [21] propose QoS estimation based on soft contracts. Soft contracts are characterized through probability distributions for QoS parameters. To yield a global contract for the composition, they use a tool called TOrQuE to unfold a composition and estimate its response time using Monte Carlo simulation. Their simulation technique assumes that the probability distribution

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

of QoS values of the elementary services are independently distributed. As discussed in this work, this assumption is often violated in practice with as consequence that their approach leads to overoptimistic or overpessimistic results.

In this paper, we present a two-step approach to predict QoS values of composite services. The first step, which is based on Petri nets, deals with **QoS aggregation** by deriving QoS values of a workflow composition from those of its participating elementary services. In contrast to related work, this technique preserves both mutual and time dependencies to allows us, in a second step, to effectively deal with the **QoS prediction** challenge. We apply time series prediction on the aggregated historical QoS to accurately predict if a composition will comply with an SLA. The accuracy of our approach is compared with the approach of Rosario et al. [21] in Section 7.2.

4. QoS considerations

4.1. QoS for elementary service

In the domain of web services, QoS parameters can be used to determine non-functional properties of the service. QoS attributes can be divided into quantitative and qualitative attributes. Examples of the latter are security and privacy. Popular quantitative attributes are response time, throughput, reputation, reliability, availability, and cost:

- Response Time (RT): the time taken to send a request and receive a response (expressed in milliseconds). The response time is the sum of the processing time and the transmission time. For short running processes they are usually of the same order. For long running processes that can take hours, days or even weeks to complete, the transmission time is usually negligible.
- Throughput (TP): the maximum requests that can be handled at a given unit in time (expressed in requests/min).
- Reputation (RP): the reputation of a service is a measure of its trustworthiness (expressed as scalar with higher value being better). The value is defined as the average ranking given to the service by end users.
- Reliability (RL): the probability that a task is satisfactorily fulfilled (expressed as a percentage). The reliability can be calculated from past data by dividing the number of successful executions by the total number of executions.
- Availability (A): the probability that a web service is available (expressed in available time/total time). It is computed by dividing the total amount of time in which a service is available through the total monitoring time. In the scope of this work, we define an available service as a service that is able to respond within a predefined time interval.
- Cost (C): the cost that a service requester has to pay for invoking a specific operation of a service (expressed in cents/request). Other pricing schemes are sometimes used such as membership fee or monthly fee.

Because the focus of this paper is on the prediction of quality of service attributes with a volatile nature, we do not consider static attributes like fixed costs such as membership fees. Also reputation is an attribute that gives a general impression about users opinions of a service, and is not meant to frequently change in time. System-level QoS attributes, such as throughput, often largely depend on hardware and computing power of the underlying infrastructure of the composite service and need to be evaluated over multiple instances. In this work, we concentrate on instance-level QoS attributes of composite services that directly relate to the QoS values of its constituent web services and moreover are non-stationary. Interesting attributes for our approach are response time, reliability, availability and cost as pay-per service. For the sake of simplicity we limit the QoS prediction algorithms in Section 6 and their evaluation in Section 7 to the response time attribute.

4.2. QoS for composite services

The QoS of a service composition is calculated based on the QoS values of its constituents. In contrast to the measurement of QoS for elementary services, composite services consist of different activities such as sequences, if-conditions, loops and

Table 5
QoS computations for composite services. $RT_i, TP_i, RP_i, RL_i, A_i$ and C_i are respectively the response time, throughput, reliability, availability and cost of the ith service. There are a total of m services which are part of a sequence, parallel execution or switch. In case of a switch the expected value is calculated where p_i is the probability that service i is executed. In case of a loop one service is executed k times.

QoS attribute	Composition patterns			
	Sequence	Parallel	Switch	Loop
Response time	$\sum_{i=1}^{m} RT_i$	$\max_{i=1}^{m}(RT_i)$	$\sum_{i=1}^{m} p_i \cdot RT_i$	$RT \cdot k$
Throughput	$\min_{i=1}^{m}(TP_i)$	$\min_{i=1}^{m}(TP_i)$	$\sum_{i=1}^{m} p_i \cdot TP_i$	TP
Reputation	$\sum_{i=1}^{m} \frac{RP_i}{m}$	$\sum_{i=1}^{m} \frac{RP_i}{m}$	$\sum_{i=1}^{m} p_i \cdot RP_i$	RP
Reliability	$\prod_{i=1}^{m} RL_i$	$\prod_{i=1}^{m} RL_i$	$\sum_{i=1}^{m} p_i \cdot RL_i$	RL^k
Availability	$\prod_{i=1}^{m} A_i$	$\prod_{i=1}^{m} A_i$	$\sum_{i=1}^{m} p_i \cdot A_i$	A^k
Cost	$\sum_{i=1}^{m} C_i$	$\sum_{i=1}^{m} C_i$	$\sum_{i=1}^{m} p_i \cdot C_i$	$C \cdot k$

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

parallel invocations. We need to take into account these different composition patterns to calculate the QoS of a composite service. Example QoS computations are summarized in Table 5. WS-BPEL elements relevant to QoS computation are simple elements as receive, reply, invoke, assign, throw, wait and complex elements like sequence, flow, if, while and foreach. Similar to Kiepuszewski et al. [13], we define a structured model that consists of four constructs that allow for recursive construction of larger workflows:

- Sequence: multiple tasks that are sequentially executed.
- Parallel execution (and-split/and-join): multiple paths that are executed concurrently en merged synchronously.
- Exclusive choice (or-split/or-join): multiple possible paths, among which only one can be executed.
- Loop: a path that is repeatedly executed a fixed number of times c.

Various standards for service composition include more constructs in addition to the four basic constructs described above. Jaeger et al. [11] summarizes the workflow patterns that cover most control constructs proposed in existing standards and products. In this paper we limit ourselves to the four basic constructs that are able to cover the most important activities offered in WS-BPEL, a popular workflow language for orchestrating web services. How the composite activities of WS-BPEL are mapped to the basic constructs will be explained in the next section. A Petri net graph and a Petri net execution time system (PNET-system) is used to reason over the workflow and to estimate its total response time.

Besides the overhead generated by the service calls, the QoS of a composite service is influenced by events internal to the orchestration. Usually, the delay caused by internal events is negligible compared to that of the service calls. This is certainly the case for medium and long running processes, which are the main targets for our approach. For further analysis, we assume that the overall delay of the orchestration depends solely on the response times of the services it calls during execution. The inclusion of internal delays is a trivial extension.

5. QoS aggregation

In the motivation, we emphasized that time related information and mutual dependencies are important to make accurate composite QoS estimates. In this section we present a QoS aggregation technique that takes both into account by generating a simulated time series of the QoS attributes of the workflow as if the composition was executed several times in the past. The quantile values will be predicted using this simulated dataset as will be explained in Section 6.

5.1. Petri net graph

Various activity-based process models have been suggested for workflow systems. We represent the workflow as a non-deterministic Petri net graph (PNG), which is a commonly used representation for workflow systems [22,18]. Compared to a structured workflow model, expressing our QoS model as a formal model offers more expressive power and is equipped with strong analysis capabilities. Furthermore, the Petri net functions as an intermediate representation that allows generalization of our approach to different workflow languages. We apply our technique in the context of WS-BPEL, however, it can be applied on any workflow language that can be represented by our Petri net representation.

Our Petrinet consists of places and transitions. Places correspond to workflow states and allow the representation of conditional execution. There are two kinds of transitions in our Petri net: timed and immediate transitions. Timed transitions represent services and the firing delay of the transition corresponds to the response time of these services. Immediate transitions are needed to enable the representation of internal events of the composite service.

A generalization of the Petri net graph we use for our analysis of WS-BPEL, is a 8-tuple $(\mathcal{P}, \mathcal{T}^1, \mathcal{T}^2, \mathcal{T}, \mathcal{D}, W^-, W^+, s_0)$, where

- $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_1}\}$ is a finite set of places. The set contains exactly one starting place (p_1) and exactly one ending place (p_{n_1}).
- $\mathcal{T}^1 = \left\{t_1^1, t_2^1, \ldots, t_{n_2}^1\right\}$ is a finite set of immediate transitions. Immediate transitions have no firing delay.
- $\mathcal{T}^2 = \left\{t_1^2, t_2^2, \ldots, t_{n_3}^2\right\}$ is a finite set of timed transitions. Timed transitions have a firing delay.
- $\mathcal{T} = \{t_1, t_2, \ldots, t_{n_4}\}$ is a finite set of transitions. All transitions are immediate or timed transitions ($\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^2$) but cannot be both ($\mathcal{T}^1 \cap \mathcal{T}^2 = \emptyset$).
- $\mathcal{D} = \{d_1, d_2, \ldots, d_{n_3}\}$ is finite set of positive real functions representing the firing delay of the corresponding timed transitions with respect to the time. $d_i(s)$ is the firing delay of t_i^2 at time s.
- W^-, W^+ are the backward and forward incidence matrices, respectively. They contain boolean values. If $W^-(i,j) = 1$, then there is an arc going from p_i to t_j. If $W^+(i,j) = 1$, then there is an arc going from t_i to p_j. If $W^-(i,j)$ or $W^+(i,j)$ equals 0, then there is no corresponding arc. Places, except for the starting and ending place, can have multiple incoming arcs ('OR-join') or multiple outgoing arcs ('OR-split'). The starting place differs in the sense it has no incoming arcs and the ending place differs in the sense it has no outgoing arcs. Timed transitions have one incoming arc and one outgoing arc. Immediate transitions can have multiple incoming arcs ('AND-join') or multiple outgoing arcs ('AND-split').

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

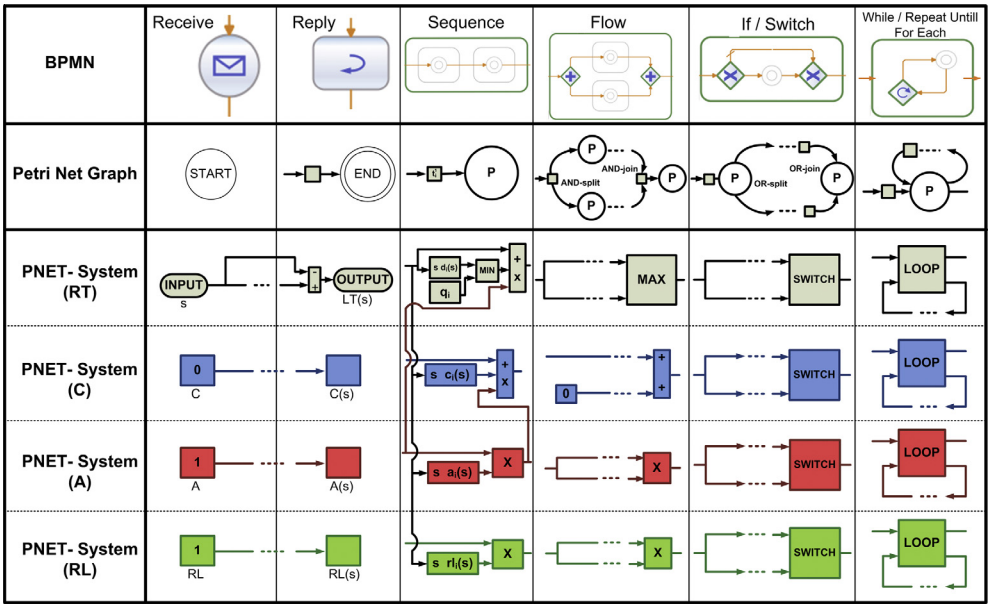

Fig. 3. Mapping: Business Process Modeling Language (BPMN) – Petri net graph – PNET-system. The gray, blue, red and green building blocks of the PNET-system correspond to respectively the response time, cost, availability and reliability. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- The initial marking of the Petri net is always one token present at the starting place. The initial time of the Petri net equals s_0. Time is continuous and can take any real value.
- An additional constraint on our Petri net is that each 'AND-split'-transition, 'OR-split'-place has to have exactly one corresponding 'AND-join'-transition, 'OR-join'-place respectively and vice versa. With corresponding we mean that all outgoing paths of the 'split'-node are disconnected until they reach the corresponding 'join'-node in which they *all* come together. The paths connecting the 'split'-node and corresponding 'join'-node are called the connecting paths.

The mapping of WS-BPEL activities, represented by their Business Process Modeling Notation (BPMN),[3] to their corresponding Petri net representation is shown in Fig. 3. If we apply this mapping on the case study introduced in Section 2.1, we get the Petri net representation illustrated in Fig. 4. The firing delays introduced by the timed transitions t_1^2 to t_7^2 correspond to the response times of the post 1, radiology, screening 1–3, report, billing and post 2 service respectively. Remark that we have truncated the Petri net graph by removing 'a place followed by an immediate transition with one incoming and one outgoing arc'. These constructs have no influence on further analysis and calculations.

Similar to Colored Petri nets, we add an extension to the elements of the net to deal with the other QoS attributes besides response time: a Petri net token is associated with 3 data values C, A and RL of type integer to hold the current aggregated cost, availability and reliability of a composition. The extension is a 4-tuple $(\mathcal{Q}, \mathcal{C}, \mathcal{A}, \mathcal{RL})$, where

- $\mathcal{Q} = \{q_1, q_2, \ldots, q_{n_3}\}$ is a finite set of positive real numbers representing the maximal allowed delay for each timed transition before it is considered as unavailable.
- $\mathcal{C} = \{c_1, c_2, \ldots, c_{n_3}\}$ is a finite set of positive real functions representing the cost of the corresponding timed transitions with respect to the time. $c_i(s)$ is the cost of t_i^2 at time s.
- $\mathcal{A} = \{a_1, a_2, \ldots, a_{n_3}\}$ is a finite set of functions where $a_i(s) \in \{0, 1\}$ represents the availability of the corresponding timed transitions with respect to the time. $a_i(s)$ is the availability of t_i^2 at time s. There is a direct relation between a_i and d_i: if $(d_i <= q_i)\{a_i = 1\}$ else $\{a_i = 0\}$
- $\mathcal{RL} = \{rl_1, rl_2, \ldots, rl_{n_3}\}$ is a finite set of functions where $rl_i \in \{0, 1\}$ represents the reliability of the corresponding timed transitions with respect to the time. $rl_i(s)$ is the reliability of t_i^2 at time s.

5.2. Petri net execution semantics

The execution of the Petri net graph is done by passing tokens from the initial marking to the end marking. These markings correspond to a token present at start place and end place respectively. An execution of the Petri net graph simulates the

[3] Business Process Modeling Notation (BPMN) Version 2.0, January 2011, OMG Specification, http://bpmn.org/.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

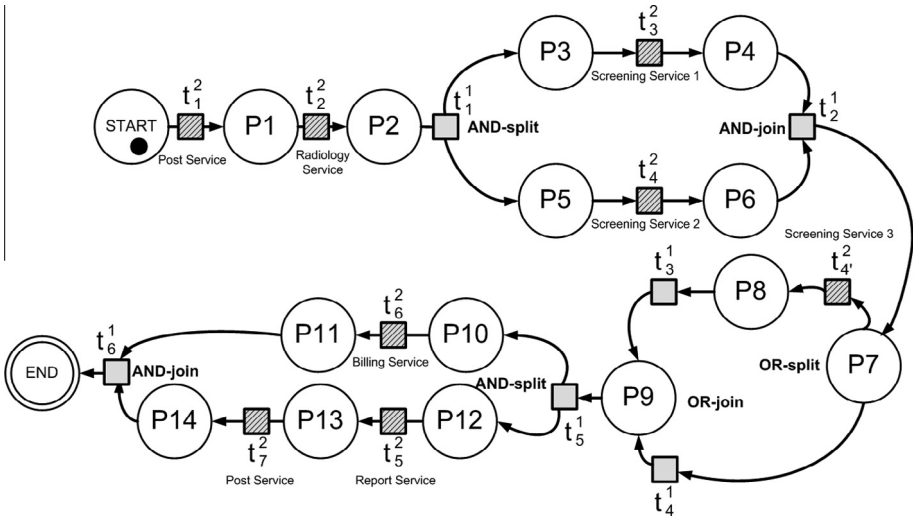

Fig. 4. Petri net graph for e-health workflow.

execution of the workflow. The execution (response) time of the workflow at time s is the time elapsed between a token present at the starting place at time s and a token reaching the ending place in the same execution cycle. During execution of the Petri net, time increases in a continuous way, starting from the initial value s_0. For each time, the following rules are executed in a non-deterministic order until no more rules can be executed:

- Rule 1: if a token is present at the incoming place of a timed transition t_i^2, then that token is consumed and the fire delay counter of that transition is activated. The counter starts counting down from $d_i(s)$ where s is the current time. As time increases, the counter value decreases with an equal amount.
- Rule 2: if an active fire delay counter of a timed transition reaches zero, then a token is generated at the outgoing place and the fire delay counter is deactivated.
- Rule 3: if tokens are present at *all* incoming places of an immediate transition, then these tokens are consumed and new tokens are generated at all outgoing places.
- Rule 4: if a token reaches the ending place, then the execution stops. Time freezes and no more rules are executed. The execution time of a Petri net graph equals the time at which the execution stops minus the initial time.

Applied on our case study, the token starts at the start place at s_0. The only applicable rule is rule 1 where the fire delay counter of the timed transition t_1^2 is activated. When the counter reaches zero at time $s_0 + d_1$, a new token is generated according to rule 2 and arrives at P1. Analogue, the token reaches P2. At P2 rule 3 is executed and new tokens are generated at both outgoing places of t_1^1. The parallel execution is done according to rule 1 followed by rule 2 for each branch in a non-deterministic order. The parallel execution ends when both tokens are consumed and a new token is generated at the immediate transition t_2^1 according to rule 3. When this token arrives at P7, both rule 1 and rule 3 can be executed corresponding to the upper and lower path respectively. Again, a rule is chosen non-deterministically. If rule 3 fires, a token is generated at the lower path and sent to the immediate transition. If rule 1 fires, a token is generated at the upper path, going to the timed transition $t_{4'}^2$. The next steps are similar to what we explained already. The execution ends after the 'AND-join', where rule 4 is applied and the token reaches its ending state. The execution time of the Petri net graph, where tokens are passed from start to end, corresponds to the simulated response time of the composite service.

To keep track of the cost, availability and reliability during the execution of the Petri net, we add the following rules as an extension to the existing ones:

- Rule 0: if a token is present at the start place, all associated data values are initialized as follows: $C \leftarrow 0, A \leftarrow 1, RL \leftarrow 1$.
- Rule 1′: if the firing delay $d_i(s)$ of a timed transition t_i^2 is above a predefined value q_i, we consider the timed transition as unavailable. The data values associated with the token are updated as follows: $A \leftarrow 0, C \leftarrow C, RL \leftarrow 0$ and the Petri net execution is halted. The Petri net execution time for the halted service then equals q_i. The response time now equals the time elapsed between a token present at the starting place and the time at which the execution is halted.
- Rule 2′: if an active fire delay counter of a timed transition t_i^2 reaches zero, then a token is generated at the outgoing place and the fire delay counter is deactivated. The data values associated with the token are updated as follows: $A \leftarrow 1, C \leftarrow C + c_i(s), RL \leftarrow RL \times rl_i(s)$.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

Taking into account these extension rules for the case study, the data values C, A and RL contain the current aggregated QoS values for the cost, availability and reliability respectively.

5.3. Petri net execution time system

Instead of simulating the above Petri net graph, we derive in this subsection a transformation of a Petri net graph into a Petri net execution time system (PNET-system). A PNET-system immediately outputs the time at which the execution of the corresponding Petri net graph stops, given the time at which the execution starts. The extended PNET-system also outputs the corresponding cost, availability and reliability of the Petri net execution. Because the Petri net is non-deterministic, the PNET-system is non-deterministic as well. An overview of the building blocks is illustrated in Fig. 6.

Definition 1. We define $S_i^-, C_i^-, A_i^-, RL_i^-$ and $S_i^+, C_i^+, A_i^+, RL_i^+$ as the values of the simulation time, cost, availability and reliability right before, respectively right after the execution of the timed transition t_i^2. The simulation time s represents the virtual time during which the Petri net is executed.

The PNET-system is generated as follows out of a Petri net graph for the different QoS attributes (see also Fig. 3 for the relationship between their constructs):

Response Time:

- Instead of calculating the execution (response) time, we are, until now, calculating the time at which the execution stops. For that reason we subtract the initial time (s_{start}) from the time at which the execution stops: $RT(s) = s - s_{start}$.
- The time at which a token is generated at the outgoing place of a timed transition t_i^2 equals the time at which a token is consumed at the incoming place added with the firing delay $d_i(s)$. A timed transition in a PNG corresponds to an addition with the delay at that time in a PNET-system. In the extended Petri net execution, any delay above the predefined value q_i is not taken into account because the workflow is then considered unavailable. Delays on an unavailable path are also not taken into account to calculate the simulated response time. The simulation time after the timed transition t_i^2 is fired at time s is thus calculated as $S_i^+ = S_i^- + \min(d_i(s), q_i) * A_i^-$.
- An 'AND-split' means all paths are executed simultaneously. The time at which the 'AND-split' is fired is copied to all outgoing places. An 'AND-split' in a PNG corresponds to a 'fork' in a PNET-system. The time at which a token is fired at the 'AND-join'-transition equals the maximum of the times on which a token is generated at the incoming places. An 'AND-join'-transition in a PNG corresponds to a 'MAX'-building block in a PNET-system.
- An 'OR-split' with corresponding 'OR-join' means a token is send into one of the connecting paths. This is equivalent to sending tokens into all connecting paths, but all but one of the former connecting paths are disconnected from the former 'OR-join'. An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.
- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

Availability & **Reliability:**

- The initial availability (reliability) is initialized to 1 (100%).
- An 'AND-split' in a PNG corresponds to a 'fork' in a PNET-system. An 'AND-join'-transition in a PNG is where two parallel paths meet. The resulting availability (reliability) is the product of availability (reliability) of both paths. An 'AND-join'-transition in a PNG corresponds to a multiplication-building block in a PNET-system.
- An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.
- The availability (reliability) at which a token is generated at the outgoing place of a timed transition equals the availability (reliability) at which a token is consumed at the incoming place multiplied with the availability (reliability) at the simulated execution time (starting time plus the time induced by the firing delays of the previous timed transition). Current simulated execution time is thus used as an input to retrieve the availability (reliability) at a specific simulated execution point.
- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

Cost:

- The initial cost is initialized to 0.
- The cost of a parallel execution equals the summation of the costs generated on all paths. To take into account the cost generated before the parallel execution, one path is connected with the previous execution path by a straight line. All other paths start counting form zero. An 'AND-join'-transition in a PNG corresponds to a summation-building block in a PNET-system. The result is a summation of all prior costs.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

- An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.
- The cost at which a token is generated at the outgoing place of a timed transition equals the cost at which a token is consumed at the incoming place plus the cost at the current simulated execution time. Also here the current simulated execution time is a necessary input to retrieve the cost. When a service is not available, the execution of the PNG stops and no further costs are made. In the PNET-system this is achieved by multiplying the cost generated by the timed transition with the aggregated availability of the system. The resulting cost after the timed transition t_i^2 that is fired at time s is thus calculated as $C_i^+ = C_i^- + c_i(s) * A_i^+$.
- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

The extended PNET-system for the e-health workflow is illustrated in Fig. 5. To keep the overview, we did not include all the building blocks as defined in Fig. 6 but used the modified functions $d_i'(s) = \min(d_i(s), q_i) * A_i^-$ and $c_i'(s) = c_i(s) * A_i^+$ instead.

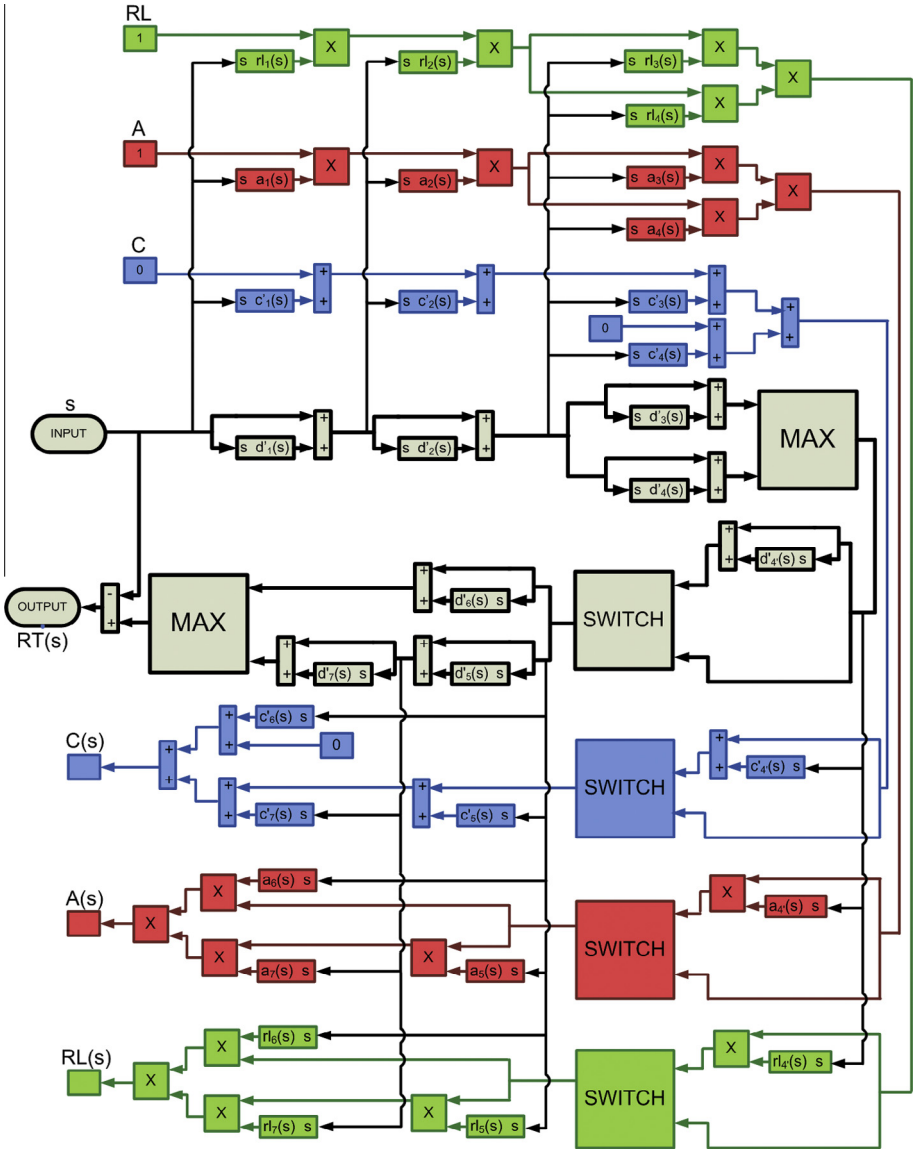

Fig. 5. Petri net execution time system for e-health workflow.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

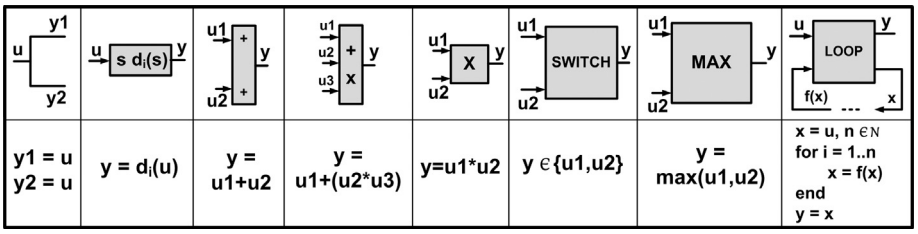

Fig. 6. Overview of the PNET-system building blocks. The top row visualizes the building blocks and the bottom row describes the input–output behavior.

5.4. From non-deterministic to deterministic

To make our approach practically usable, we need to find a way to simulate a non-deterministic system using a deterministic algorithm. To allow ex-ante estimation of QoS values, we need to make assumptions on how many times a loop will be executed and which path will be followed after a conditional execution. This is not always trivial because the number of loop executions or the value of a condition is often only known at run-time. Possible strategies for resolving the non-deterministic constraints by making deterministic assumptions are:

- Assume the worst-case scenario. In case of a loop, this means to use the maximum number of times it can be executed in practice. For a conditional execution, the worst-case path depends on the QoS attribute. The worst response time is on the path that takes the longest time to execute. This can be modeled by replacing the 'OR-split' and 'OR-join' by an 'AND-split' and 'AND-join' (parallel execution with synchronization) in the Petri net graph and replacing the 'SWITCH' by a 'MAX' building block in the PNET-system. The worst-case cost is on the most expensive path and for availability (reliability), it is the least available (reliable) path. In practice, one can simulate the QoS values for all paths and take the worst values for each attribute. For a more general approach to worst case execution analysis, we refer to specialized literature in this domain [19].
- Use a probabilistic model by assigning probabilities to the number of times a loop is executed or to each path that is implied by a condition. In practice, this strategy can be realized by doing the simulation for all paths considered and assign probabilities to the resulting QoS values. An important remark here is that the resulting values cannot simply be added after multiplication with their corresponding probabilities if one wants to estimate quantile values.

5.5. Example simulated QoS calculation

In this section, we explain more in detail how we generate a composite time series from individual time series of the constituting services using the PNET-system. Suppose we have monitored the response times for several consequent time periods of the 8 services used in the mammography workflow as shown in Fig. 7. We generate a virtual time series by simulating the execution of the composite service according to the PNET-system discussed in the previous subsection. The inputs are fixed time steps in the past. For example, if the workflow would be executed at time 0.8, we can see that the execution of the first service (post service) takes approximately 1.31 time units. This implies that the second service (radiology service)

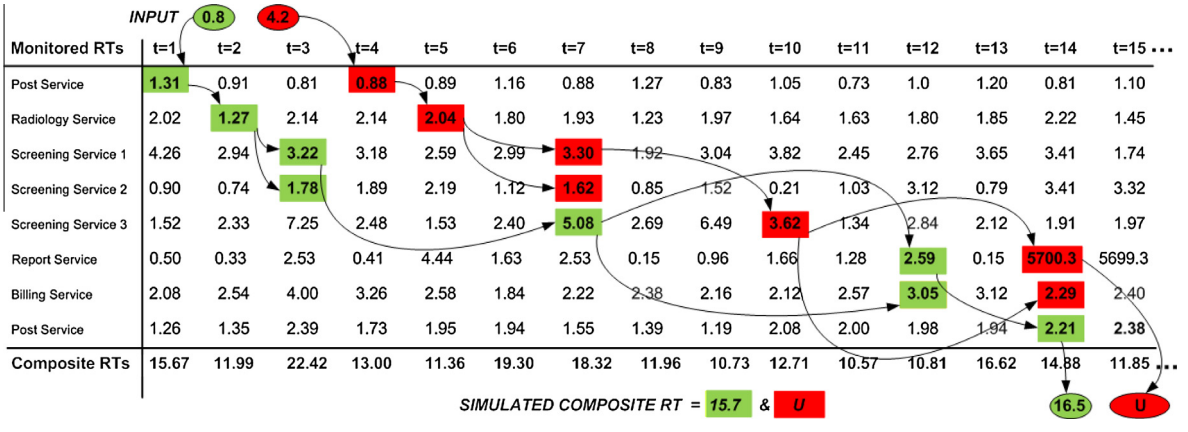

Fig. 7. Example simulated RT calculation for the e-health workflow. The response times that are highlighted in green are the response times corresponding to a composition that would have been executed successfully. The response times that are highlighted in red are the response times corresponding to a composition that would not have been executed successfully.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

will be executed at time 2.11 (sum of 0.8 and 1.31). The closest monitored result is at $t = 2$ with a response time of 1.27 units. The resulting execution time is 3.38 (sum of 2.11 and 1.27). At this point, the two screening services S1 and S2 will be executed in parallel with corresponding RTs of 3.22 and 1.78. Because this is a parallel execution where the workflow has to wait for the slowest service to finish, we will arrive at the next service at time 6,61 (maximum ending time of SS1 and SS2). The other calculations are similar. For the *if condition*, we have to make a deterministic assumption concerning the path that will be chosen. In practice, in most of the cases of a mammography screening the results of SS1 and SS2 will probably match, meaning that a third opinion is not necessary. This implies that the upper path of the conditional execution in Fig. 5 rarely will be chosen. Nevertheless, the safest strategy is to calculate the worst case scenario. This means we take into account the path that generates the maximum response time of all the paths that could be implied by the *if condition*. Worst case calculations of the response time of the composite service using input times (s_{start}) of 0.8 and 4.2 are

$$RT_{0.8} = 0.8 + 1.31 + 1.27 + \max(3.22; 1.78) + \max(5.08; 0) + \max(2.59 + 2.21; 3.05) - 0.8 = 15.7$$
$$RT_{4.2} = 4.2 + 0.88 + 2.04 + \max(3.30; 1.62) + \max(3.62; 0) + \max(5700.3 + U; 2.29) - 4.2 = U.$$

Remark that for the second calculation, the report service is unavailable and takes 5700.3 s to recover from failure. For services further in the execution chain, QoS values cannot be retrieved because no data is available that far in the future. An unavailable value is marked as 'Unknown (U)' and has the following properties: $0 \times U = 0; 1 \times U = U; \max(U, x) = U; \min(U, x) = U; U + x = U$. In our extended calculation below, we tackle this problem by taking into account the other QoS attributes and their interrelations.

To simulate the composite QoS values for cost, availability and reliability, we need their monitored values for all participating services at the times they are actually executed. Table 6 shows some fictive QoS values we use to illustrate our calculations. When a service on the path of execution is not available, the aggregated availability and reliability will become 0 and the costs and response times of the remaining services will not be taken into account due to its multiplication with the aggregated availability in the PNET-system. Worst case calculation for input times of 0.8 and 4.2 are

$$\begin{aligned} RT_{0.8} = {} & 0.8 + \min(1.31, 2.10) \times 1 + \min(1.27, 6.14) \times 1 + \max(\min(3.22, 6.32) \times 1; \min(1.78, 20.15) \times 1) \\ & + \max(\min(5.08, 15.95) \times 1; 0) + \max(\min(2.59, 49.35) \times 1 + \min(2.21, 5.18) \times 1; \min(3.05, 4.82) \times 1) \\ & - 0.8 = 15.7 \end{aligned}$$

$$A_{0.8} = 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1$$

$$C_{0.8} = 6.72 \times 1 + 203.30 \times 1 + 110.37 \times 1 + 103.17 \times 1 + 125.81 \times 1 + 6.71 \times 1 + 9.30 \times 1 + 9.88 \times 1 = 575.26$$

$$RL_{0.8} = 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1$$

$$\begin{aligned} RT_{4.2} = {} & 4.2 + \min(0.88, 2.10) \times 1 + \min(2.04, 6.14) \times 1 + \max(\min(3.30, 6.32) \times 1; \min(1.62, 20.15) \times 1) \\ & + \max(\min(3.62, 15.95) \times 1; 0) + \max(\min(5700.3, 49.35) \times 1 + \min(U, 5.18) \times 0; \min(2.29, 4.82) \times 1) \\ & - 4.2 = 63.6 \end{aligned}$$

Table 6
Monitored QoS values and worst case composite QoS calculations for the 8 services used in the mammography workflow. The maximal allowed delay before the service is considered unavailable is denoted as q. The time period at which the service is executed is denoted as t and s represents the simulation time when the workflow would be executed. For each services there are two rows: the first row corresponds to the composition that would have been executed successfully and the second row corresponds to the composition that would not have been executed successfully.

Monitored QoS	q	t	RT	C	A	RL
Post service	2.10	1	1.31	6.72	1	1
		4	0.88	5.77	1	1
Radiology service	6.14	2	1.27	203.30	1	1
		5	2.04	235.33	1	1
Screening service 1	6.32	3	3.22	110.37	1	1
		7	3.30	110.89	1	1
Screening service 2	20.15	3	1.78	103.17	1	1
		7	1.62	102.62	1	1
Screening service 3	15.95	7	5.08	125.81	1	1
		10	3.62	113.10	1	1
Report service	49.35	12	2.59	6.71	1	1
		14	5700.3	8.57	0	0
Billing service	4.82	12	3.05	9.30	1	1
		14	2.29	5.24	1	0
Post service	5.15	14	2.21	9.88	1	1
		–	–	–	–	–
Composite QoS	$s = 0.8$		15.67	575.26	1	1
	$s = 4.2$		63.6	572.95	0	0

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

$$A_{4.2} = 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 0 \times 1 \times 1 = 0$$

$$C_{4.2} = 5.77 \times 1 + 235.33 \times 1 + 110.89 \times 1 + 102.62 \times 1 + 113.10 \times 1 + 8.57 \times 0 + 5.24 \times 1 + U \times 0 = 572.95$$

$$RL_{4.2} = 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 0 \times 0 \times U = 0.$$

Using this technique to calculate a time series for the composite service, we can apply the same prediction algorithm to estimate the QoS for elementary and composite services.

5.6. Discussion

Related works on QoS prediction typically make predictions for the individual services [4,5,21] and combine these predictions using QoS aggregation strategies such as Monte Carlo simulation [21] or probabilistic composition rules [10]. In our approach, we simulate a complete time series for the composite service, as it was executed several times in the past, and apply prediction on the resulting composite time series. Existing aggregation techniques have the following disadvantages compared to our simulation approach:

- Quantile values of QoS attributes of the individual service are not sufficient to calculate a realistic global quantile value for the composite service. One needs to predict the probability density for the QoS attributes of each individual service.
- The quantile values depend not only on the probability density of the QoS values of the individual services but on the dependencies between these QoS values as well. These mutual dependencies need to be modeled explicitly. Our strategy implicitly takes them into account as discussed below.
- QoS of individual services are often modeled as distributions that do not preserve information on how the QoS evolves in time. We believe that these time dependencies are valuable to make accurate estimations of future QoS values by means of time series prediction.

We elaborate on how mutual and time dependencies are implicitly taken into account by our aggregation approach. The example in Section 2.4 illustrates that aggregation causes distorted results when mutual dependencies (due to time dependencies) are not taken into account. Our approach, however, gives the correct response times for the composite service (CS) equivalent to those shown in Table 3. The starting time of the post service (PS) is aggregated by the PNET-system after the execution of the screening service (SS) to generate the composite time series and thus the weekend will never be counted twice. In case of mutual dependencies due to resource dependencies we get similar results. The result of our simulation approach is a time series for the composite QoS. The time information of this series can be exploited, as illustrated in Section 2.3, to make more accurate predictions for the future composite QoS. How this time series prediction can be done will be discussed in the next section.

6. QoS prediction

The goal we want to achieve with our estimation algorithm is to predict if the chance that a certain SLO will be violated is larger than a predefined value. This section is organized as follows. In Section 6.1 we define the problem we try to solve by discussing two performance indicators we want to minimize out-of-sample. A kernel-based batch learning algorithm designed to minimize the first performance indicator is discussed in Section 6.2. Section 6.3 explains an online algorithm designed to minimize the second performance indicator. Finally Section 6.4 combines both of the previous algorithms such that a good performance is achieved on both indicators.

6.1. Performance indicators

We want to check whether a service has a chance of at least τ to have a response time smaller than the maximal response time $f_{i,max}$. This can be achieved by first obtaining the confidence interval $[0, f_\tau(x_i)]$ such that the chance the response time

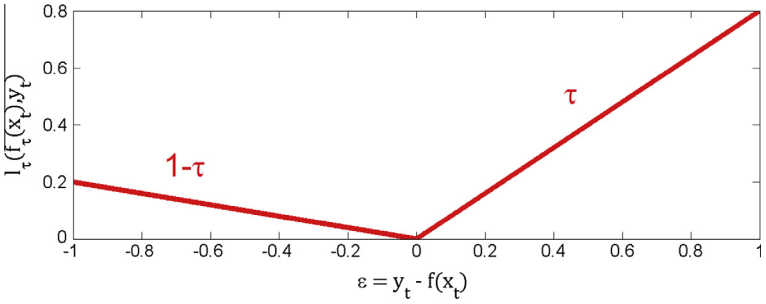

Fig. 8. Pinball loss function used for quantile estimation. On the figure τ equals 0.8.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

belongs to this interval equals τ. The service is then selected if $f_\tau(x_i) \leqslant f_{i,max}$. The true conditional quantile value is denoted as $\mu_\tau(x_i)$ and satisfies

$$\begin{aligned} &P(y_i < \mu_\tau(x_i)) \leqslant \tau \\ &P(y_i \leqslant \mu_\tau(x_i)) \geqslant \tau. \end{aligned} \tag{3}$$

For simplicity reasons we assume the quantile value is unique (which is not necessary true). The performance indicators explained in this section try to quantify how good the produced $f_\tau(x_i)$ performs. The produced quantile value that is expected to perform the best, should be the true quantile value.

6.1.1. Performance indicator I

To be able compare different quantile estimators we have to quantify how good the estimated quantile value approximates the true quantile value. For a location estimator the mean value minimizes the square error and the median value minimizes the absolute value of the error. Similarly it can be shown the quantile value minimizes the following pinball loss function [14] (Fig. 8):

$$l_\tau(y_i - f_\tau(x_i)) = \begin{cases} \tau(y_i - f_\tau(x_i)), & y_i - f_\tau(x_i) \geqslant 0 \\ (\tau - 1)(y_i - f_\tau(x_i)), & y_i - f_\tau(x_i) < 0. \end{cases} \tag{4}$$

This loss is however not unique in the sense that there exists other loss-functions that have an optimum in the quantile value as well, such as

$$l_{\tau,\log} = l_\tau(\log(y_i) - \log(f_\tau(x_i))). \tag{5}$$

It is moreover possible that one estimator performs better according to one of these loss functions (e.g. l_τ) and another estimator performs better according to another one (e.g. $l_{\tau,\log}$). That is why, in this section, we further specify the problem of estimating the quantile value to a problem of minimizing a well chosen cost. Two types of error can occur in our applications:

- Type I error: a (composite) service is rejected to execute a customer's request in which the actual response time is smaller than the maximal response time ($y_i < f_{i,max}$).
- Type II error: a (composite) service is accepted to execute a customer's request in which the actual response time is larger than the maximal response time ($y_i > f_{i,max}$).

For the first performance indicator we assign costs to both errors: the cost of a type I error equals $1 - \tau$ and the cost of a type II error equals τ. In Theorem 1 and Corollary 1 we will show that the expected cost can be minimized if one knows the true conditional quantile value.

Theorem 1. *If the cost of a type I error equals $1 - \tau$ and the cost of a type II error equals τ, then the expected conditional cost for accepting is lower than the expected conditional cost for rejecting if and only if $f_{i,max} > \mu_\tau(x_i)$.*

Proof 1. The expected conditional cost for accepting a service equals

$$E(\text{cost}|\text{accept}, x_i) = P(y_i > f_{i,max}|x_i)\tau, \tag{6}$$

where $P(y_i > f_{i,max}|x_i)$ equals the probability that $y_i > f_{i,max}$, and the expected conditional cost for rejecting a service equals

$$E(\text{cost}|\text{reject}, x_i) = P(y_i < f_{i,max}|x_i)(1 - \tau). \tag{7}$$

Accepting is better than rejecting if and only if

$$\begin{aligned} E(\text{cost}|\text{accept}, x_i) - E(\text{cost}|\text{reject}, x_i) < 0 &\iff P(y_i > f_{i,max}|x_i)\tau - P(y_i < f_{i,max}|x_i)(1 - \tau) < 0 \\ &\iff P(y_i > f_{i,max}|x_i) < 1 - \tau \\ &\iff f_{i,max} > \mu_\tau(x_i). \quad \square \end{aligned} \tag{8}$$

Corollary 1. *If $f_\tau(x_i)$ equals the true conditional quantile value $\mu_\tau(x_i)$ and a service is accepted if and only if $f_\tau(x_i) < f_{i,max}$, then the expected conditional cost is minimized for all possible values of $f_{i,max}$.*

A type I error occurs when $y_i < f_{i,max} < f_\tau(x_i)$ and a type II error occurs when $f_\tau(x_i) < f_{i,max} < y_i$. Given a test set only containing the true response times y_t, we cannot calculate the cost because we have no values for $f_{i,max}$. We need extra assumptions to handle the uncertainty over $f_{i,max}$. The first performance indicator (PI_1) will be defined as the cumulative expected cost, given these assumptions.

Theorem 2. *If we receive one service request per time step, if the probability on a certain f_{max} is time independent and uniformly distributed in an interval $[f^-, f^+]$ with f^-, f^+ finite and if all $y_t, f_\tau(x_t)$ belong to this interval, then the expected cost, given y_t and $f_\tau(x_t)$, becomes proportional to the pinball loss (l_τ)*

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

$$E(\text{cost}|y_t, f_\tau(x_t)) \sim l_\tau(y_t - f_\tau(x_t)). \tag{9}$$

Proof 2. When $f_\tau(x_t) > y_t$ a type I error (with cost $1-\tau$) can occur and when $y_t < f_\tau(x_t)$ a type II error (with cost τ) can occur. Because f_{max} is uniformly distributed, the probability on these errors becomes proportional to the distance between $f_\tau(x_t)$ and y_t and the expected cost becomes proportional to the pinball loss. □

Corollary 2. *Given the same assumptions as in Theorem 2, the first performance indicator becomes*

$$\text{PI}_1(\mathcal{T}, f_\tau) = \sum_{t=1}^{n_{test}} l_\tau(y_{t,test} - f_\tau(x_{t,test})). \tag{10}$$

A disadvantage of using the pinball loss as performance measure is that the loss of one datapoint is not bounded (it can become infinite), despite the fact that a datapoint can only cause one error of type I or II. This is caused by the assumption that all y_t and $f_\tau(x_t)$ belong to the interval $[f^-, f^+]$ and the interval $[f^-, f^+]$ is thus not a priori determined and can become arbitrary large.

Theorem 3. *If we receive one service request per time step and if the probability on a certain f_{max}, denoted as P, is time independent and a priori determined, then the first performance indicator (PI_1) equals the following cumulative expected cost*

$$\text{PI}_1(\mathcal{T}, f_\tau, F) = \sum_{t=1}^{n_{test}} l_\tau\big(F(y_{t,test}) - F(f_\tau(x_{t,test}))\big) \tag{11}$$

where F is the cumulative distribution function of P

$$F(y) = \int_0^y P(f_{t,max}) df_{t,max}. \tag{12}$$

Proof 3. If $y_{t,test} \geqslant f_\tau(x_{t,test})$, then an error of type II can occur with probability $F(y_{t,test}) - F(f_\tau(x_{t,test}))$. The expected cost becomes $\tau(F(y_{t,test}) - F(f_\tau(x_{t,test})))$. If $y_{t,test} < f_\tau(x_{t,test})$, then an error of type I can occur with probability $F(f_\tau(x_{t,test})) - F(y_{t,test})$. The expected cost becomes $(1-\tau)(F(f_\tau(x_{t,test})) - F(y_{t,test}))$. □

The influence of one data point on PI_1 is bounded because

$$l_\tau\big(F(y_{t,test}) - F(f_\tau(x_{t,test}))\big) \leqslant \max(\tau, 1-\tau). \tag{13}$$

Different probability distributions for f_{max} cause different costs and in practice the performance indicator should equal the cumulative expected cost in which the chosen probability distribution of f_{max} matches the true probability distribution as good as possible. In our experiments we will use the performance indicator as defined in Corollary 2 in which the influence of one datapoint is unbounded because we have no data that allows us to make a reasonable estimate of $P(f_{max})$.

6.1.2. Performance indicator II

In practice costs are not always linear in the number of errors. It is possible that causing more violations than agreed invokes huge fines while causing fewer violations than agreed does not cause equally large profits. For that reason it is important that the number of violations does not exceed the agreed number of violations too much. Constraining the frequency of violations not to exceed $1-\tau$ is in our opinion too severe because the test set contains only a sample of the total population and it is possible the frequency of violations of this sample exceeds $1-\tau$ despite the frequency of violations of the entire population to be smaller or equal to $1-\tau$. That is why we choose performance indicator II to express the probability that the frequency of violations of the test set is larger than or equal to the actual number of violations in the test set, given the frequency of violations of the entire population equals $1-\tau$.

Theorem 4. *Assuming the expected number of violations equals $1-\tau$ and assuming the data is independent and identically distributed (i.i.d.), the probability of observing at least $n_{v,test}$ violations on a test set where $n_{req,test}$ service requests are accepted becomes*

$$P(n_v \geqslant n_{v,test}|\tau, n_{req,test}) = \sum_{i=n_{v,test}}^{n_{req,test}} (1-\tau)^i \tau^{(n_{req,test}-i)} \binom{n_{req,test}}{i} \tag{14}$$

where $\binom{n}{i}$ is a binomial coefficient and where the number of violations are counted as follows

$$n_{v,test} = \sum_{i=1}^{n_{req,test}} \text{viol}(y_{i,test}, f_{i,max}) \text{ where } \text{viol}(y_{i,test}, f_{i,max}) = \begin{cases} 0, & y_{t,test} \leqslant f_{i,max} \\ 1, & y_{t,test} > f_{i,max}. \end{cases} \tag{15}$$

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

Proof 4. The discrete probability distribution of the number of violations n_v in a sequence of $n_{req,test}$ independent experiments, each of which yields a violation with probability $1-\tau$, equals a binomial distribution. The probability on $n_{v,test}$ or more violations given a binomial distribution is expressed in Eq. (14). □

To be able to measure the number of violations, we need test data containing times on which service requests are send together with the corresponding maximal response times. In case no such data is available, we have to make extra assumptions.

Corollary 3. *Under the extra assumptions of one service request per time step and 100% acceptance with $f_{i,max}$ equal to $f_\tau(x_i)$ (the latter assumption implies the worst case value of $f_{i,max}$ such that the service is accepted), the probability of observing at least $n_{v,test}$ violations is denoted as PI_2 and equals*

$$\mathrm{PI}_2(\mathcal{T}, f_\tau) = \sum_{i=n_{v,test}}^{n_{test}} (1-\tau)^i \tau^{(n_{test}-i)} \binom{n_{test}}{i} \tag{16}$$

where $\binom{n}{i}$ is a binomial coefficient and where the number of violations are counted as follows

$$n_{v,test} = \sum_{t=1}^{n_{test}} \mathrm{viol}(y_{t,test}, f_\tau(x_{t,test})) \text{ where } \mathrm{viol}(y_{t,test}, f_\tau(x_{t,test})) = \begin{cases} 0, & y_{t,test} \leqslant f_\tau(x_{t,test}) \\ 1, & y_{t,test} > f_\tau(x_{t,test}). \end{cases} \tag{17}$$

The true quantile value of an estimator is defined as

$$\tau_f(f_\tau) = P(y_{t,test} < f_\tau(x_{t,test})). \tag{18}$$

Given the violations are counted as in Eq. (17), $1-\tau_f(f_\tau)$ is equal to the expected number of observed violations. The smaller PI_2, the more unlikely the observed number of violations are caused by an estimator with true quantile value larger than or equal to τ. A high PI_2, however, does not imply having a conditional quantile estimator. An unconditional, constant function can perform very well on this performance indicator, but will, luckily, perform poorly on PI_1.

6.2. Kernel-based quantile estimation

In this Section we will explain why the kernel-based quantile estimator discussed in [25] suits our problem. The first algorithm we propose is designed to minimize PI_1 and thus the following *expected risk*

$$R[f_\tau] = \int l_\tau(y - f_\tau(x)) dP(x,y). \tag{19}$$

The probability density function $P(x,y)$ is unknown. We have only access to training data. A naive way of minimizing the *expected risk* is minimizing the *empirical risk*

$$R_{emp}[f_\tau, S] = \frac{1}{n}\sum_{t=1}^{n_{tr}} l_\tau(y_{t,tr} - f_\tau(x_{t,tr})). \tag{20}$$

When the hypothesis space $\mathcal{H}$ to which f_τ belongs is very rich, this can lead to overfitting. For that reason we will minimize the *regularized risk*

$$R_{reg}[f_\tau, S] = R_{emp}[f_\tau, S] + \frac{\lambda}{2}\|w\|_2^2 = \frac{1}{n}\sum_{t=1}^{n_{tr}} l_\tau(y_{t,tr} - f_\tau(x_{t,tr})) + \frac{\lambda}{2}\|w\|_2^2 \tag{21}$$

where λ is a regularization parameter that must be positive. We assume f_τ can be written as

$$f_\tau(x) = w^T\varphi(x) + b \tag{22}$$

where w is the weight vector, φ maps the input space into a feature space (which can be infinite dimensional) and b is the constant offset. Minimizing the *regularized risk* can be written as the following optimization problem

$$\min_{w,b,\xi,\xi^*} \sum_{t=1}^{n_{tr}} \tau\xi_t + (1-\tau)\xi_t^* + \frac{1}{2}\lambda\|w\|_2^2 \text{ subject to} \begin{cases} y_{t,tr} - w^T\varphi(x_{t,tr}) - b \leqslant \xi_t, & t = 1,\ldots,n_{tr} \\ w^T\varphi(x_{t,tr}) + b - y_{t,tr} \leqslant \xi_t^*, & t = 1,\ldots,n_{tr} \\ \xi_t, \xi_t^* \geqslant 0, & t = 1,\ldots,n_{tr}. \end{cases} \tag{23}$$

Because $\varphi(x)$ can be huge or even infinite dimensional, directly solving this optimization problem can become very expensive. The dual problem, after applying the kernel trick, becomes

$$\min_\alpha \frac{1}{2}\lambda\alpha^T\Omega\alpha - \alpha^T y \text{ subject to } \begin{cases} \frac{\tau}{\lambda} \geqslant \alpha_t \geqslant \frac{\tau-1}{\lambda}, & t = 1,\ldots,n_{tr} \\ \sum_{t=1}^{n} \alpha_t = 0 \end{cases} \tag{24}$$

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

where $k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j), \Omega \in \mathbb{R}^{n_{tr} \times n_{tr}}$ is the kernel matrix defined as $\Omega_{i,j} = k(x_{i,tr}, x_{j,tr}), \alpha = [\alpha_{1,tr}, \ldots, \alpha_{n_{tr},tr}]^T$ and $y = [y_{1,tr}, \ldots, y_{n_{tr},tr}]^T$. This optimization problem is a quadratic programming (QP) problem. The kernel trick ($k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$) makes explicit calculation of the mapping $\varphi(x)$ into a possible infinite dimensional feature space no longer necessary. The function f_τ can now be expressed as

$$f_\tau(x) = \sum_{t=1}^{n_{tr}} \alpha_t k(x_{t,tr}, x) + b. \tag{25}$$

The constant offset b can be found using the fact that the following holds $f_\tau(x_{t,tr}) = y_{t,tr}$ for α_t belonging to the open interval $]\frac{\tau-1}{\lambda}, \frac{\tau}{\lambda}[$. The pseudo-code for the algorithm explained in this section is shown in Algorithm 1.

Algorithm 1. Given the training set $\mathcal{S}$, the targeted quantile value τ, a kernel k and the regularization parameter λ, find α and b which are parameters of the predicting function f_τ.

for $t_1 = 1 \rightarrow n_{tr}$ **do**
  for $t_2 = 1 \rightarrow n_{tr}$ **do**
    $\Omega_{t_1,t_2} \leftarrow k(x_{t_1,tr}, x_{t_2,tr})$
  end for
end for
$\hat{\alpha} \leftarrow \arg\ \min_\alpha \frac{1}{2}\lambda\alpha^T\Omega\alpha - \alpha^T y_{tr}$

$$\text{subject to } \begin{cases} \frac{\tau}{\lambda} \geqslant \alpha_t \geqslant \frac{\tau-1}{\lambda}, & t = 1, \ldots, n_{tr} \\ \sum_{t=1}^{n} \alpha_t = 0 \end{cases}$$

for $t = 1 \rightarrow n_{tr}$ **do**
  if $\hat{\alpha}_t \in\ ]\frac{\tau-1}{\lambda}, \frac{\tau}{\lambda}[$ **then**
    $b \leftarrow y_{t,tr} - \sum_{t_1=1}^{n_{tr}} \hat{\alpha}_{t_1} k(x_{t_1,tr}, x_{t,test})$
  end if
end for

6.3. Online adaptation of an unconditional quantile estimator

The second algorithm we propose is designed to minimize PI_2. It is a simple online algorithm that is updated as follows

$$f_{\tau,t} = \begin{cases} f_{\tau,start}, & t = 1 \\ f_{\tau,t-1} + \eta\tau, & t > 1, y_{t-1} > f_{\tau,t-1} \\ f_{\tau,t-1} + \eta(\tau - 1), & t > 1, y_{t-1} < f_{\tau,t-1} \\ f_{\tau,t-1}, & t > 1, y_{t-1} = f_{\tau,t-1}. \end{cases} \tag{26}$$

where η is the learning rate which must be strictly positive. The start value $f_{\tau,start}$ can be set equal to 0, equal to the first data point, or equal to the quantile value of a number of datapoints. Datapoints used to determine $f_{\tau,start}$ should not be used for the online updating. The number of times y_{t_1} exceeds f_{τ,t_1} for t_1 going from 1 to t is denoted as v_t. We can now express $f_{\tau,t+1}$ as

$$f_{\tau,t+1} = f_{\tau,1} + \eta(\tau v_t + (\tau - 1)(t - v_t)) = f_{\tau,1} + \eta t\left(\frac{v_t}{t} - (1 - \tau)\right). \tag{27}$$

Conversely, v_t can be expressed in function of $f_{\tau,t+1}$ as follows

$$v_t = \frac{f_{\tau,t+1} - f_{\tau,1}}{\eta} + (1 - \tau)t. \tag{28}$$

Theorem 5. *Assuming there exists an interval $[y_{min}, y_{max}]$ with y_{min}, y_{max} finite such that $f_{\tau,start}$ and all y_t belong to this interval, the frequency of violations converges to $1 - \tau$ for t going to ∞*

$$\lim_{t \to \infty} \frac{v_t}{t} = 1 - \tau. \tag{29}$$

Proof 5. Because all y_t belong to the interval $[y_{min}, y_{max}]$, all $f_{\tau,t}$ belong to the interval $[y_{min} + \eta(\tau - 1), y_{max} + \eta\tau]$ and

$$|f_{\tau,t+1} - f_{\tau,1}| \leqslant C \Rightarrow \eta t \left|\frac{v_t}{t} - (1-\tau)\right| \leqslant C \Rightarrow \left|\frac{v_t}{t} - (1-\tau)\right| \leqslant \frac{C}{\eta t} \tag{30}$$

where $C = y_{max} - y_{min} + \eta \max(\tau, 1-\tau)$. Expression (30) shows the distance between $\frac{v_t}{t}$ and $(1-\tau)$ converges to zero for t going to ∞. □

Theorem 6. *Assuming all y_t belong to the interval $[y_{min}, y_{max}]$ with y_{min}, y_{max} finite, the performance indicator PI_2 of the test set converges to 0.5 for $t \to \infty$.*

Proof 6. Suppose the random variable X_t follows a binomial distribution which yields a violation with probability $1-\tau$ in a sequence containing t datapoints, then PI_2 expresses the probability the number of violations and thus X_t is at least v_t. The mean and the standard deviation of X_t equal

$$\mu_{X,t} = t(1-\tau) \tag{31}$$

$$\sigma_{X,t} = \sqrt{t\tau(1-\tau)}. \tag{32}$$

From Eq. (30) we can now conclude

$$\frac{|v_t - \mu_{X,t}|}{\sigma_{X,t}} \leqslant \frac{C}{\eta\sqrt{t\tau(1-\tau)}}, \tag{33}$$

which implies

$$\lim_{t\to\infty} \frac{v_t - \mu_{X,t}}{\sigma_{X,t}} = 0. \tag{34}$$

According to the central limit theorem, the cumulative distribution function (CDF) of X_t converges pointwise to the CDF of a normal distribution with mean $\mu_{X,t}$ and standard deviation $\sigma_{X,t}$ for n going to ∞. Because, as n approaches ∞, v_t is distanced 0 standard deviations from $\mu_{X,t}$, the CDF of X_t in v_t equals 0.5 and PI_2 becomes 0.5. □

The pseudo-code for the algorithm explained in this section is shown in Algorithm 2.

Algorithm 2. Given x_{test}, y_{test}, η and $f_{\tau,start}$, calculate $f_{\tau,t}(x_{t,test})$ and update $f_{\tau,t}$ for $t = 1, \ldots, n_{test}$.

```
f_{τ,1} ← f_{τ,start}
for t = 1 → n_test do
    if y_{t,test} > f_{τ,t} then
        f_{τ,t+1} ← f_{τ,t} + τη
    else if y_{t,test} < f_{τ,t} then
        f_{τ,t+1} ← f_{τ,t} + (τ − 1)η
    else
        f_{τ,t+1} ← f_{τ,t}
    end if
end for
```

6.4. Kernel-based quantile estimation with online adaptation of the constant offset

In this section we will present the final algorithm that combines the algorithms explained in the previous two sections. Additionally we define the input vectors and explain how the data is preprocessed. The final algorithm consists of three stages.

- Preprocessing phase: in this phase the inputs and corresponding outputs are generated from the response times RT_t. We will predict the logarithm of response times instead of the response times themselves because this avoids our algorithm to make predictions below zero. The input vectors can be past response times (autoregressive), the quantile values of sets of past response times or the time at which the service starts. The best choice for the input vectors depends on the specific dataset one wants to make predictions on.
- Training phase: secondly we optimize α and b on the training set using Algorithm 1 to predict the logarithm of the response time.
- Online updating phase: finally we try to predict $\log(y_t) - f_\tau(x_t)$ with a function denoted as $d_{\tau,t}$ using Algorithm 2. The optimal $d_{\tau,start}$ can be determined using a validation set or can be set to 0. We add d_t to $f_\tau(x_t)$ to make the final prediction. The latter is equivalent to updating the constant offset b_t such that b_t equals $b + d_t$.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

Table 7
Toy dataset used to illustrate the kernel-based quantile estimator with online adaptation of the constant offset. The time, input vector and output vector are denoted respectively as t, x_t and y_t. The kernel-based quantile estimator without and with online adaptation are denoted respectively as f and g_t. The function g_t equals f plus the online adaptation function d_t. The quantile parameter τ equals 0.75. For simplicity we use fixed hyperparameters: the regularization parameter λ equals 0, the kernel is linear and the learning rate η equals 0.1. The first four datapoints are training data and the last four data points are test data. The initial value of the online adaptation parameter (d_5) is set to 0.

t	Training data				Test data			
	1	2	3	4	5	6	7	8
$x_t(=t)$	1	2	3	4	5	6	7	8
y_t	0.9	2.3	2.9	4.1	5.1	6.2	6.9	7.8
$f(x_t)$	1.4	2.3	3.2	4.1	5.0	5.9	6.8	7.7
d_t	\	\	\	\	0	0.075	0.15	0.125
$g_t(x_t)$	\	\	\	\	5.0	5.975	6.95	7.825

This final algorithm performs well on both PI_1 and PI_2. To further clarify how this algorithm works, we will illustrate it using the toy example shown in Table 7. First the estimator f is learned using the training data and d_5 is set to zero. At time 5 the predicted value $g_5(5) = 5.0$ is smaller than the true value $y_t = 5.1$ and therefore the online adaptation parameter is increased as follows $d_6 = d_5 + \tau \cdot \eta = 0 + 0.75 \cdot 0.1$. At time 6 the predicted value equal to 5.975 is still smaller than the true value equal to 6.2 and thus $d_7 = d_6 + \tau \cdot \eta$. At time 7 the predicted value is larger than the true value and $d_8 = d_7 + (\tau - 1) \cdot \eta$.

7. Experimental evaluation

We limit the experimental evaluation to predicting the response time. In a first scenario we predict the response time of an individual service, in a second scenario we predict the response time of an entire workflow.

7.1. Prediction for individual services

Algorithm 3. Given the true response times RT_t for $t = 1, \ldots, n + L$ generate the input vectors x_t and the output values y_t for $t = 1, \ldots, n$. RT_t is the true response time at time t, x_t is the input vector at time $t + L$ and y_t is the output vector at time $t + L$. The function $quantile(\tau, \mathcal{X})$ expresses the τ-quantile value of the set $\mathcal{X}$.

for $t = 1 \rightarrow n$ **do**
$q_{1,t} \leftarrow quantile\left(\tau, \{RT_i\}_{i=t}^{t+\frac{L}{4}-1}\right)$
$q_{2,t} \leftarrow quantile\left(\tau, \{RT_i\}_{i=t+\frac{L}{4}}^{t+\frac{L}{2}-1}\right)$
$q_{3,t} \leftarrow quantile\left(\tau, \{RT_i\}_{i=t+\frac{L}{2}}^{t+\frac{3L}{4}-1}\right)$
$q_{4,t} \leftarrow quantile\left(\tau, \{RT_i\}_{i=t+\frac{3L}{4}}^{t+L-1}\right)$
$x_t \leftarrow [\log(q_{1,t}), \log(q_{2,t}), \log(q_{3,t}), \log(q_{4,t})]^T$
$y_t \leftarrow \log(RT_{t+L})$
end for

The kernel-based quantile estimator (KQ) explained in Section 6.2 and the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) explained in Section 6.4 will be evaluated using a real-life data set discussed in Appendix B: the Movie Dataset. We also compare the performance of our algorithms to a sliding window approach, which is used in related work [21].

The kernel we use is a RBF-kernel. The input vectors are quantile values of sets of past response time as shown in Algorithm 3. KQ has three hyperparameters: the regularization parameter λ, the kernel parameter σ and the lag L. KQOA has an extra parameter: the online learning rate η. The training set consists of the first 700 datapoints and will be used to find the optimal α and b, given the hyperparameters. The validation set consists of the next 300 datapoints and will be used to optimize the hyperparameters λ, σ and L. The learning rate η of the second algorithm allows adaptation to unexpected events and should not be chosen too small. Unexpected behavior that might happen in the test set might not happen in the validation set. We have chosen η as big as possible such that the performance according to performance indicator I on the validation set does not increase with more than 2%.

When the optimal hyperparameters are found, the algorithm is retrained on both the training and the validation set. The performance will then be evaluated using the test set which contains the remaining 4000 datapoints. We compare

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

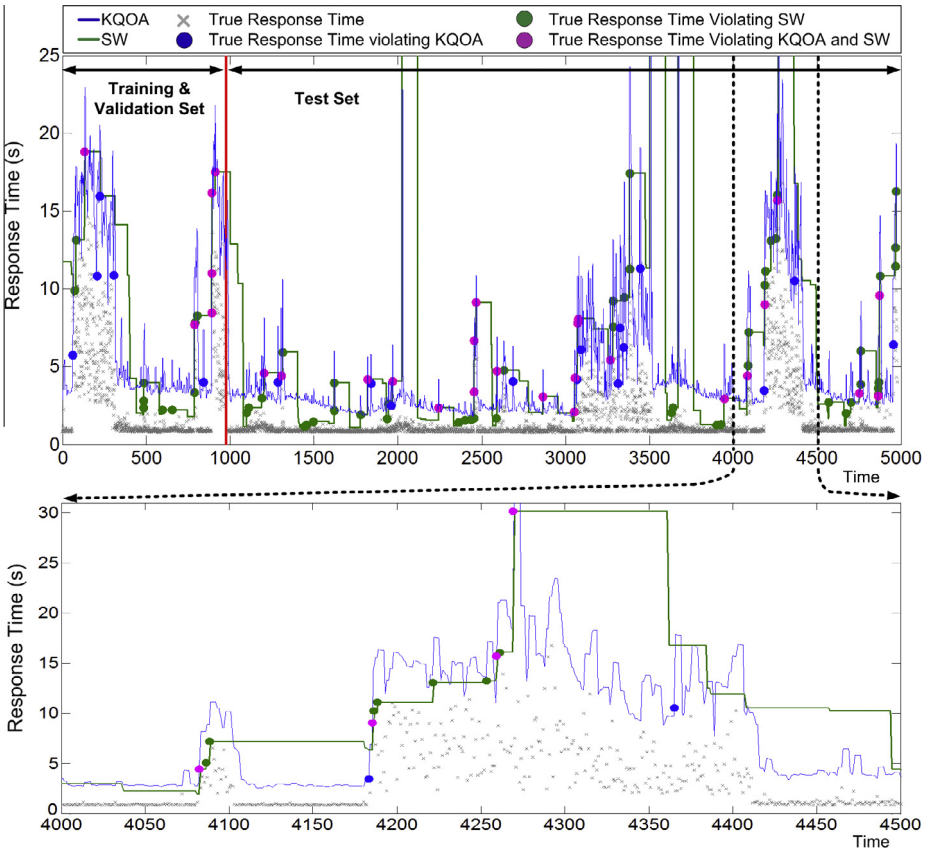

Fig. 9. Quantile estimation for the Movie Dataset using the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) and the sliding window estimator (SW). The upper figure shows the results for data point 0–5000. The lower figure zooms in on data point 4000–4500.

Table 8
Comparison of the performance of the Kernel-based Quantile Estimator with Online Adaptation of Constant Offset (KQOA) and the Sliding Window Quantile Estimator (SWQ) on the Movie Dataset (a) and the Movie Dataset with an increase of 15s in response time of 50 randomly selected test set datapoints (b). PI_1 equals the cumulative pinball loss, $PI_2(\%)$ is the probability an estimator with true quantile value 99% causes equal or more violations and $f_{v,test}$ is the number of violations in the test set per 100 datapoints.

	PI_1	$PI_2(\%)$	$f_{v,test}$
(a) Movie dataset			
KQOA	0.0635	58.5	0.98
KQ	0.0637	99.6	0.63
SWQ	0.0974	0.0	1.85
		$PI_2(\%)$	$f_{v,test}$
(b) Adjusted movie dataset			
KQOA		6.9	1.25
KQ		0.0	1.75
SWQ		0.0	2.95

our algorithms to a sliding window quantile estimator with windowsize w. The latter estimator uses the τ-quantile value of the lasts w datapoints as τ-quantile estimation for the next datapoint. The parameter w is optimized using the first 1000 datapoints as training set and evaluated using the next 4000 datapoints as test set. The KQOA and SW quantile estimations for the Movie Dataset are shown in Fig. 9. The lower figure zooms in on the area between data point 4000 and 4500. The performances of the estimators for a quantile value of 99% are summarized in Table 8(a). According to PI_1 the expected cost using the kernel-based algorithms is approximately 30% lower compared to the sliding window algorithm. According to PI_2 the probability an estimator with true quantile value 99% causes equal or more violations than the estimator is quite big for the kernel-based algorithms and almost non-existing for the sliding window estimator.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

We can conclude our algorithms performs significantly better than the sliding window estimator on both PI_1 and PI_2. Without online adaptation, however, we cannot guarantee that PI_2 will not converge to zero. With online adaptation, given certain assumptions, we can. To show the effect of the online adaptation, we increased the response time of 50 randomly selected datapoints of the test set by 15 s. This causes the KQ algorithm to perform bad because these spikes only appear in the test set and are thereby not learned during the training period. In Table 8(b), we can see that for this experiment only KQOA has a value for PI_2 that differs significantly from zero. The optimal hyperparameters for our algorithms are: $L = 24, \sigma = 500, \lambda = 0.005$ and $\eta = 0.1$. The optimal window size for the sliding window quantile estimator equals 92.

7.2. Prediction for composite services

For predicting the response time of composite services, we need to calculate the simulated response times according to the technique explained in Section 5. We use the e-health case study described in Section 2.1, where each elementary service has a response time according to simulated data for a long-running process. Rosario et al. [21] validated the use of certain families of distributions and performed their best fit on measured data real-life web services. They observed that the Gamma and the log–logistic distributions were a reasonably good fit for the response times.

To simulate data for the elementary services, we will assume the number of working hours needed to complete a task is modeled as log–logistic distributed random variables. The log–logistic distribution is similar in shape to the log–normal distribution but has heavier tails. The probability density function depends on α and β:

$$f(x;\alpha,\beta) = \frac{\left(\frac{\beta}{\alpha}\right)\left(\frac{x}{\alpha}\right)^{\beta-1}}{\left[1+\left(\frac{x}{\alpha}\right)^{\beta}\right]^2}. \tag{35}$$

Some of the services in our case study require human interaction and thus we assume the completion time of a task is subject to working hours of the service. Depending on the service, a working day starts and ends at a specific hour and are open or closed on Saturdays and Sundays. An overview of our assumptions is shown in Table 9. Outside the range of working hours the progress of the task is freezed. For example, if a working day starts at 8 h and ends at 16 h, then a task taking 9 working hours to complete that starts at 10 h on Monday will end at 11 h on Tuesday. The response time of 25 h consist of 6 working hours within the same day, 16 non-working hours and 3 working hours the next day. We assume monitoring is done every hour. The generated response times for all elementary services of the e-health workflow are shown in Fig. 10. The graph at the bottom represents the resulting composite response time, calculated using the PNET-system described in Section 5.

We will compare the Kernel-Based Quantile estimator (KQ) explained in Section 6.2 and the Kernel-Based Quantile estimator with Online Adaptation of the constant offset (KQOA) explained in Section 6.4 to the probabilistic approach using Bootstrap-Based Simulations explained in the paper of Rosario [21]. In this work, they use bootstrap based Monte Carlo (BBMC) simulations to estimate a composite QoS value by repeatedly drawing QoS values from the probability distributions of the constituting elementary services and composing them using soft contract rules. The training set consists of the first 4 weeks, the validation set (used to optimize hyperparameters) of the next 4 weeks and the test set consists of the last 8 weeks. After optimizing the hyperparameters the algorithm is retrained on both the validation and training set. BBMC uses past response time as input which has the problem that recent executions of individual services may not have ended yet. Their response time is thus unknown. We solve this by sliding the window backwards in time until all response times within and before the window are known. As shown in Algorithm 4, we take the time (in h) within a week as input vector for KQ and KQOA, because we a priori know there is weekly seasonality. The KQOA and BBMC quantile estimations for the e-health application are shown in Fig. 11. Table 10 shows KQ and KQOA perform better than BBMC and shows the number of violations for BBMC is very low.

The estimated quantile value of BBMC overestimates the true quantile value which can be explained as follows: the BBMC does not take into account the mutual dependencies due to time dependencies as we illustrated in Section 2.4. For some services the progress freezes during the weekends. The response time of these services on Friday evening will be very high. The Monte Carlo simulation combines random samples of the services response times, without considering if this combination

Table 9
Parameter assumptions for elementary services of e-health case study.

Services	Parameters				
	Start (h)	End (h)	Weekend	α	β
Post 1	8	15	Closed	log(10)	1/8
Radiology	9	18	Closed	log(14)	1/10
Screening 1	8	20	Saturday	log(12)	1/9
Screening 2	8	22	Closed	log(12)	1/7
Screening 3	8	20	Saturday	log(15)	1/7
Report	0	24	Open	log(2)	1/10
Billing	0	24	Open	log(5)	1/4
Post 2	8	15	Closed	log(10)	1/8

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

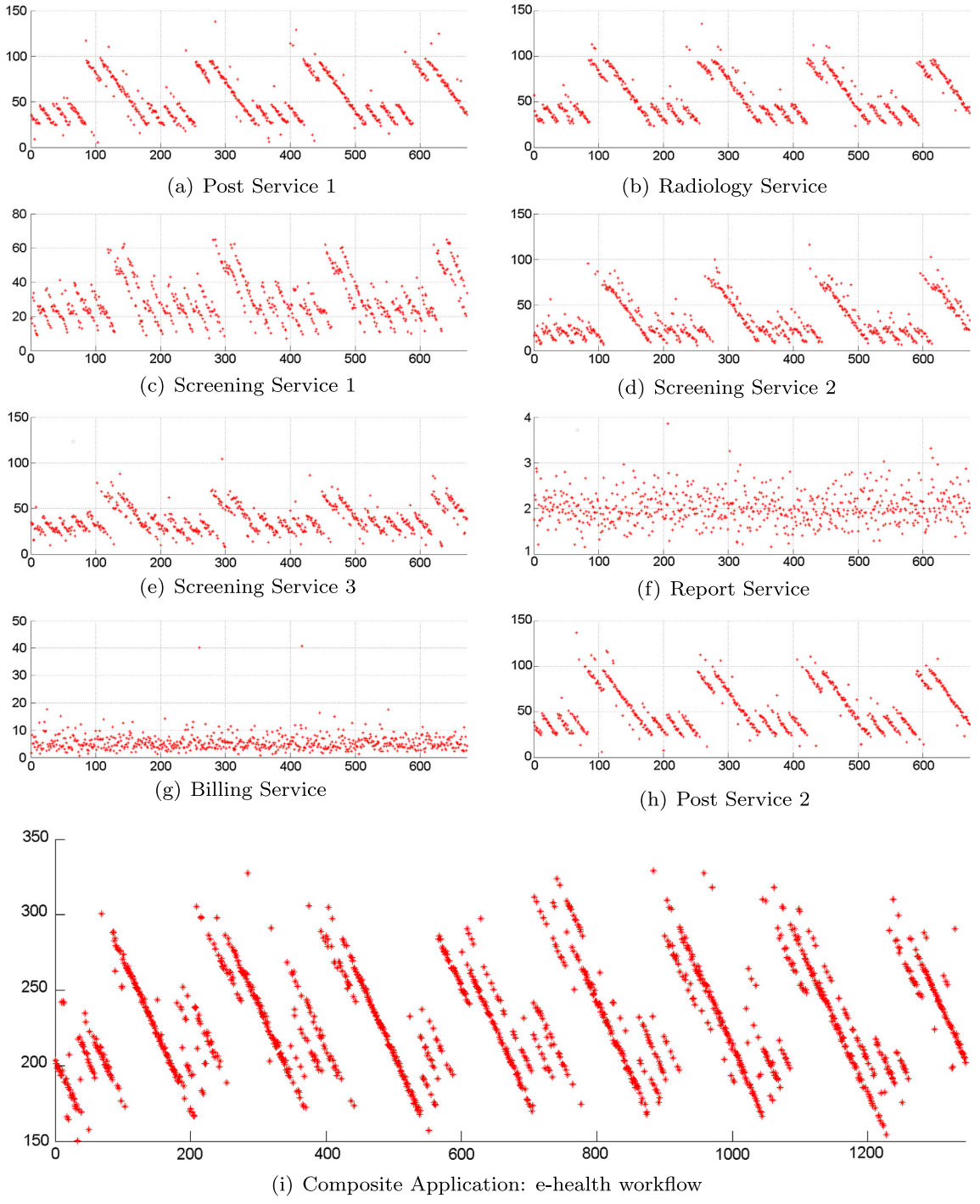

Fig. 10. Overview of response times of elementary service (a–h) and resulting simulated response time of composite service (i) for e-health case-study (horizontal axis: time (in h) – vertical axis: response time (in h)).

can occur in practice. The worst case scenario using Monte Carlo thus becomes worse than the 'actual' worst case scenario which causes the estimated 99%-quantile value to be too high. The approach used in KQ and KQOA of first composing the response times as explained in Section 5 and then estimating the quantile values avoids this problem. Another reason for the poor performance of BBMC is that it cannot learn the seasonality patterns.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

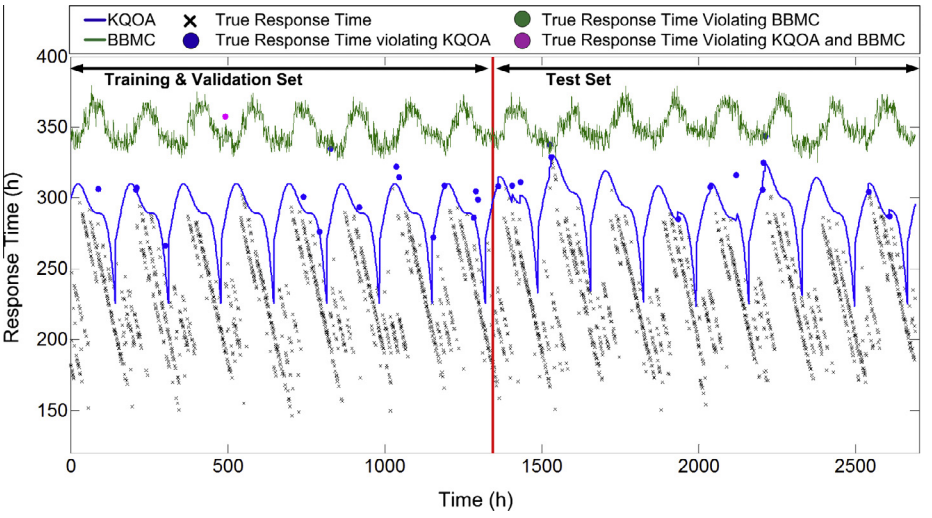

Fig. 11. Quantile estimation for the e-health application using the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) and the bootstrap based Monte Carlo estimator (BBMC).

Table 10
Comparison of the performance of the Kernel-based Quantile Estimator (KQ), Kernel-based Quantile estimator with Online Adaptation of Constant Offset (KQOA) and the probabilistic approach using Boostrap-Based Simulations (BBMC) on the E-Health Dataset. PI_1 equals the cumulative pinball loss, $PI_2(\%)$ is the probability an estimator with true quantile value 99% causes equal or more violations and $f_{v,test}$ is the number of violations in the test set per 100 datapoints. The chosen hyperparameters for KQOA and/or KQ are: $\sigma = 50, \lambda = 0.01$ and $\eta = 0.02$. The chosen window-size for BBMC equals 300.

	PI_1	$PI_2(\%)$	$f_{v,test}$
KQOA	0.795	69.1	0.89
KQ	0.857	3.3	1.56
BBMC	1.212	100.0	0.0

Table 11
Confusion matrix.

	Accepted	Rejected
No Violation	True Positive	Type I Error
Violation	Type II Error	True Negative

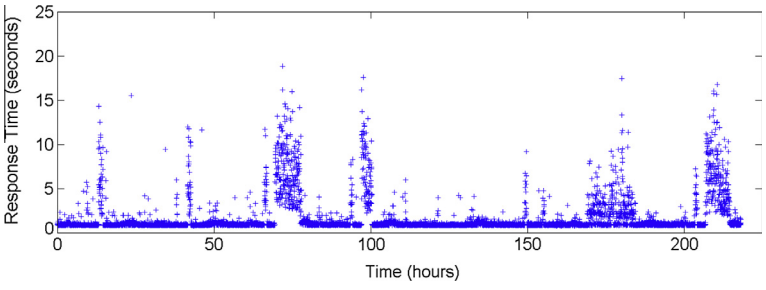

Fig. 12. Time-varying RTs of online web service.

Algorithm 4. Given the true response times RT_t for $t = 1, \ldots, n$ generate the input vectors x_t and the output values y_t for $t = 1, \ldots, n$.

for $t = 1 \rightarrow n$ **do**
  $x_t \leftarrow t \bmod 24 \cdot 7$
  $y_t \leftarrow RT_t$
end for

8. Conclusion

In this paper we developed a two-step approach to accurately predict QoS quantiles for composite services. The first step, which is based on Petri nets, derives simulated QoS-values of a workflow composition from those of its constituent elementary services. In a second step we focus on the response time and try to predict future response times of composite services based on the simulated response times using a kernel-based quantile estimator with online adaptation of the constant offset. The latter algorithm has a batch and an online learning phase. During the batch learning phase it tries to minimize, in a regularized manner and given certain assumptions, the occurrence of two types of errors: a type I error occurs when a service is rejected in which the real response time is smaller than the agreed response time and a type II error occurs when a service is accepted in which the real response time exceeds the agreed response time. The online learning phase is necessary to assure that the probability an estimator with true quantile value equal to the agreed quantile value causes equal or more violations, converges to 50% for the number of test datapoints going to infinity. The latter implies the number of times the true response time exceeds the estimated response time converges to the agreed quantile value.

We evaluated our prediction algorithm on one elementary service, an automated service using real-life web service data, and on one composite service, a service with human interaction using simulated data. On the first dataset our algorithm caused a loss 30% lower than a sliding window estimator and on the second dataset our algorithm significantly outperformed the probabilistic approach using Boostrap-Based Simulations (BBMC) explained in the paper of Rosario et al. [21].

Despite the promising results our approach has some limitations:

- When making small changes to the composition, the kernel-based quantile estimator needs to be retrained completely. This makes it expensive to use this algorithm as a part of a composition selection procedure. It can however still be used to verify whether a chosen composition satisfies the SLO. Once the algorithm is trained, however, doing prediction on the chosen composition for future time steps is relatively cheap because the training does not need to be redone.
- The prediction algorithms are most effective for processes with variable QoS that changes according to recognizable patterns. Typically this is the case for long-running processes that require human interaction. When the QoS attributes of candidate services are close to random, the use of prediction algorithms is not effective and will only slow down the composition process. The first step, where composite QoS-values of a workflow composition are derived from those of its elementary services, however, is suitable for all types of processes, and can easily be combined with other estimation techniques.

9. Future work

Our future work includes a practical implementation of QoS-aware service composition using the algorithms presented in this paper. Currently we are evaluating our approach in the context of Cloud Computing. Software as a service (SaaS), sometimes referred to as "Software on Demand", is an emerging mechanism of releasing software applications to customers. From a technical perspective, an import issue in "software on demand" is the customization for the SaaS application to serve multiple tenants, each having their own service level requirements. Our goal is to have a practically efficient solution to address customization of business processes with a focus on run-time assurance of service quality and service level agreements. For the implementation we use the framework discussed in [8]. The framework extends the WS-BPEL language with policy-based reconfiguration capabilities. The framework is based on the Model-View-Controller (MVC) pattern and is implemented in Ruby On Rails (RoR). To achieve accurate estimates of the response times of the individual services and extend them to an estimate for the composite service, we will use the techniques described in this work. The logic to efficiently select appropriate services among available services will be implemented using the reconfiguration support of the framework. A challenge here is to implement an efficient selection algorithm that takes into account the stochastic behavior of the QoS estimates of candidate services by integrating the QoS estimation techniques presented in this work.

Acknowledgements

Research supported by: IBBT, Research Council KUL: GOA/10/09 MaNet, PFV/10/002 (OPTEC), several PhD/postdoc & fellow grants; Flemish Government: IOF: IOF/KP/SCORES4CHEM; FWO: PhD/postdoc grants, projects: G.0588.09 (Brain-machine), G.0377.09 (Mechatronics MPC), G.0377.12 (Structured systems); IWT: Ph.D. Grants, projects: SBO LeCoPro, SBO

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

Climaqs, SBO POM, EUROSTARS SMART: iMinds 2013; Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012–2017); EU: FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), ERC AdG A-DATADRIVE-B; COST: Action ICO806: IntelliCIS. The scientific responsibility is assumed by its authors.

Appendix A

Theorem 7. *Optimization problem*

$$\min_{f_\tau \in \mathcal{H}} \quad P(\text{Type I error}|f_\tau)(1-\tau) + P(\text{Type II error}|f_\tau)\tau \tag{A.1}$$

$$\text{such that } P(\text{Type II error}|\text{accepted}, f_\tau) \leqslant 1-\tau \tag{A.2}$$

has the same solutions as optimization problem

$$\min_{f_\tau \in \mathcal{H}} \quad P(\text{Type II error}|\text{accepted}, f_\tau) + P(\text{rejected}|f_\tau)((1-\tau) - P(\text{Type II error}|\text{accepted}, f_\tau)) \tag{A.3}$$

$$\text{such that } (1-\tau) - P(\text{Type II error}|\text{accepted}, f_\tau) \geqslant 0. \tag{A.4}$$

Proof 7. For simplicity we do the proof for the optimization problem without constraint (A.2), nevertheless it still holds with the constraint. First we rewrite Eq. (A.1) as

$$\min_{f_\tau \in \mathcal{H}} \quad (P(\text{Type I error}|f_\tau) - P(\text{Type II error}|f_\tau))(1-\tau) + P(\text{Type II error}|f_\tau).$$

As can be derived from the confusion matrix in Table 11, the following equation hold

$$P(\text{Type I error}|f_\tau) + P(\text{Violation}|f_\tau) = P(\text{Rejected}|f_\tau) + P(\text{Type II error}|f_\tau). \tag{A.5}$$

The number of violations is a constant in the sense that it does not depend on whether we accept the request or not, therefore

$$P(\text{Violation}|f_\tau) = P(\text{Violation}). \tag{A.6}$$

After applying Eqs. (A.5) and (A.6), the optimization problem becomes

$$\min_{f_\tau \in \mathcal{H}} \quad (P(\text{Rejected}|f_\tau) - P(\text{Violation}))(1-\tau) + P(\text{Type II error}|f_\tau).$$

The term $P(\text{Violation})(1-\tau)$ is a constant for a given data set and can therefore be omitted from the optimization problem without affecting the optimum

$$\min_{f_\tau \in \mathcal{H}} P(\text{Rejected}|f_\tau)(1-\tau) + P(\text{Type II error}|f_\tau).$$

If we replace

$$P(\text{Type II error}|f_\tau) = P(\text{Type II error}|\textit{Accepted}, f_\tau)(1 - P(\textit{Rejected}|f_\tau))$$

in the optimization problem, we get the statement we had to proof: optimization problem (A.3). □

Appendix B

A current problem for experiments regarding QoS of real web services is the lack of available datasets. For our analyses, we found no usable time series on Quality of Web Service attributes. Service providers usually only publish average values for their services. The QWS dataset[4] of Al-Masri and Mahmoud. [2] includes measurements of 9 QoS attributes for 2500 real web services. Each service was tested over a 10-min period for three consecutive days. However, only the average QoS values are publicly available. Another public dataset is WS-DREAM[5] which offers real invocation info on 100 web services by using 150 distributed computer nodes located all over the world. The dataset contains data on consecutive invocations of the services but is limited to 100 time series datapoints per service, which is not sufficient for our experiments. There is also no labeling on the time span of the different invocations of a service. Both datasets are restricted to short-running automated web services.

To cope with the data problem, we have collected real time series data for short-running online services ourselves. We used Web Inject,[6] a free client-side monitoring tool for automated testing of web applications and web services. The tool allows

[4] http://www.uoguelph.ca/~qmahmoud/qws/index.html.
[5] http://www.wsdream.net/wsdream/.
[6] http://www.webinject.org/.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063

to send soap requests to web services to analyse their response time and fault counts. We monitored a set of 8 popular online web services over a 2-min period for 7 consecutive days. Fig. 12 illustrates the response time of a web service that allows a client to retrieve information on movies and theaters in the US. We can observe that a QoS attribute like response time can be very dynamic in time.

References

[1] R. Aggarwal, K. Verma, J. Miller, W. Milnor, Constraint driven web service composition in meteor-s, in: Proceedings of the IEEE International Conference on Services Computing, SCC'04, IEEE Computer Society, Shanghai, China, 2004, pp. 23–30.
[2] E. Al-Masri, Q.H. Mahmoud, Investigating web services on the world wide web, in: Proceedings of the 17th International Conference on World Wide Web, WWW'08, ACM, Beijing, China, 2008, pp. 795–804.
[3] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, An approach for qos-aware service composition based on genetic algorithms, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO'05, ACM, Washington DC, USA, 2005, pp. 1069–1075.
[4] J. Cardoso, Quality of Service and Semantic Composition of Workflows, Ph.D. thesis, University of Georgia, 2002.
[5] J. Cardoso, A. Sheth, J. Miller, J. Arnold, K. Kochut, Quality of service for workflows and web service processes, J. Web Seman. 1 (2004) 281–308.
[6] B.-J. Chen, M.-W. Chang, C.-J. lin, Load forecasting using support vector machines: a study on eunite competition 2001, IEEE Trans. Power Syst. 19 (2004) 1821–1830.
[7] A. Christmann, I. Steinwart, Consistency of kernel-based quantile regression, Appl. Stoch. Mod. Bus. Indust. 24 (2008) 171–183.
[8] K. Geebelen, E. Kulikowski, E. Truyen, W. Joosen, A mvc framework for policy-based adaptation of workflow processes: a case study on confidentiality, in: Proceedings of the IEEE International Conference on Web Services, ICWS'10, Santa Clara Marriott, CA, USA, pp. 401–408.
[9] J. Harney, P. Doshi, Speeding up adaptation of web service compositions using expiration times, in: Proceedings of the 16th International Conference on World Wide Web, WWW'07, ACM, Banff, Alberta, Canada, 2007, pp. 1023–1032.
[10] S.-Y. Hwang, H. Wang, J. Tang, J. Srivastava, A probabilistic approach to modeling and estimating the qos of web-services-based workflows, Inform. Sci. 177 (2007) 5484–5503.
[11] M.C. Jaeger, G. Rojec-Goldmann, G. Muhl, Qos aggregation for web service composition using workflow patterns, in: Proceedings of the 8th IEEE International Conference on Enterprise Distributed Object Computing, EDOC'04, IEEE, Monterey, CA, USA, 2004, pp. 149–159.
[12] M.C. Jaeger, G. Mühl, S. Golze, Qos-aware composition of web services: an evaluation of selection algorithms, in: Proceedings of the Confederated International Conference on the Move to Meaningful Internet Systems, OTM'05, Springer Verlag, Berlin, Germany, 2005, pp. 646–661.
[13] B. Kiepuszewski, A.H.M. t. Hofstede, C. Bussler, On structured workflow modelling, in: Proceedings of the 12th International Conference on Advanced Information Systems Engineering, CAiSE'00, Springer Verlag, Gdansk, Poland, 2000, pp. 431–445.
[14] R. Koenker, J. Bassett, Gilbert, egression quantiles, Econometrica 46 (1978) 33–50.
[15] P. Leitner, A. Michlmayr, F. Rosenberg, S. Dustdar, Monitoring, prediction and prevention of sla violations in composite services, in: Proceedings of the IEEE International Conference on Web Services, ICWS'10, IEEE Computer Society, Miami, FL, USA, 2010, pp. 369–376.
[16] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: benchmarking of state-of-the-art techniques, Patt. Recog. 36 (2003) 2271–2285.
[17] D. Mukherjee, P. Jalote, M. Gowri Nanda, Determining qos of ws-bpel compositions, in: Proceedings of the 6th International Conference on Service-Oriented Computing, ICSOC'08, Springer Verlag, Durban, South Africa, 2008, pp. 378–393.
[18] S. Narayanan, S.A. McIlraith, Simulation, verification and automated composition of web services, in: Proceedings of the 11th International Conference on World Wide Web, WWW'02, ACM, Honolulu, HI, USA, 2002, pp. 77–88.
[19] P. Puschner, A. Burns, Guest editorial: a review of worst-case execution-time analysis, Real-Time Syst. 18 (2000) 115–128.
[20] O.F. Rana, M. Warnier, T.B. Quillinan, F. Brazier, D. Cojocarasu, Managing violations in service level agreements, Grid Middle. Serv. (2008) 349–358.
[21] S. Rosario, A. Benveniste, S. Haar, C. Jard, Probabilistic qos and soft contracts for transaction-based web services orchestrations, IEEE Trans. Serv. Comput. 1 (2008) 187–200.
[22] K. Salimifard, M. Wright, Petri net-based modelling of workflow systems: an overview, Euro. J. Operat. Res. 134 (2001) 664–676.
[23] A. Strunk, Qos-aware service composition: a survey, in: IEEE 8th European Conference on Web Services, ECOWS'10, IEEE Computer Society, Ayia Napa, Syprus, 2010, pp. 67–74.
[24] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Pub. Co., Singapore, 2002.
[25] I. Takeuchi, Q.V. Le, T.D. Sears, A.J. Smola, Nonparametric quantile estimation, J. Mach. Learn. Res. 7 (2006) 1231–1264.
[26] W.M.P. Van Der Aalst, A.H.M. Ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distrib. Parall. Datab. 14 (2003) 5–51.
[27] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
[28] W. Wiesemann, R. Hochreiter, D. Kuhn, A stochastic programming approach for qos-aware service composition, in: Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, CCGRID'08, IEEE Computer Society, Lyon, France, 2008, pp. 226–233.
[29] H. Xianglan, L. Yangguang, X. Bin, Z. Gang, A survey on qos-aware dynamic web service selection, in: Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM'11, Wuhan, China, pp. 1–5.
[30] T. Yu, Y. Zhang, K.-J. Lin, Efficient algorithms for web services selection with end-to-end qos constraints, ACM Trans. Web 1 (2007).
[31] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Qos-aware middleware for web services composition, IEEE Trans. Softw. Eng. 30 (2004) 311–327.

Please cite this article in press as: D. Geebelen et al., QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2013.12.063